Федеральное государственное автономное образовательное учреждение высшего образования
“Санкт-Петербургский политехнический университет Петра Великого”

На правах рукописи

Трифонов Петр Владимирович

Методы построения и декодирования многочленных кодов

05.13.17 – Теоретические основы информатики

ДИССЕРТАЦИЯ
на соискание ученой степени
доктора технических наук

Санкт-Петербург – 2018

Оглавление

Введение . . . 8

Глава 1. Корректирующие коды и методы их декодирования . . . 18

1.1. Технические требования к средствам помехоустойчивого кодирования . . . 18

1.1.1. Системы передачи информации . . . 18

1.1.2. Системы хранения данных . . . 19

1.2. Основные понятия теории кодирования . . . 19

1.2.1. Теоретико-информационные основы . . . 20

1.2.2. Линейные блоковые коды . . . 21

1.2.3. Вероятность ошибки декодирования . . . 22

1.2.4. Списочное декодирование . . . 22

1.2.5. Каналы передачи информации . . . 23

1.3. Обобщенные каскадные коды . . . 25

1.3.1. Описание конструкции . . . 25

1.3.2. Декодирование . . . 26

1.3.3. Выбор компонентных кодов . . . 27

1.3.3.1. Правило пропускных способностей . . . 27

1.3.3.2. Правило сбалансированных минимальных расстояний . . . 28

1.3.3.3. Правило экспоненты случайного кодирования . . . 29

1.3.3.4. Правило предельных вычислительных скоростей . . . 30

1.3.3.5. Правило равных вероятностей ошибки . . . 30

1.3.3.6. Коды для списочного декодирования . . . 30

1.4. Полярные коды . . . 31

1.4.1. Поляризация канала . . . 31

1.4.2. Ядра . . . 33

1.4.2.1. Ядро Арикана . . . 33

1.4.2.2. Ядро БЧХ . . . 35

1.4.2.3. Ядро Рида-Соломона . . . 36

1.4.3. Полярные коды как обобщенные каскадные . . . 36

1.4.4. Списочный метод последовательного исключения . . . 37

1.4.4.1. Алгоритм Тала-Варди . . . 37

1.4.4.2. Аппроксимация весовой функции . . . 37

1.4.4.3. Вероятность ошибки . . . 38
1.5. Коды Рида-Соломона . . . 39
1.5.1. Определения . . . 39
1.5.2. Декодирование кодов Рида-Соломона . . . 39
1.5.3. Мономиальные упорядочения . . . 42
1.5.4. Метод Гурусвами-Судана списочного декодирования кодов Рида-Соломона . . . 42
1.5.4.1. Интерполяция . . . 43
1.5.4.2. Поиск функциональных корней . . . 45
1.5.4.3. Перекодирование . . . 45
1.5.5. Метод By списочного декодирования кодов Рида-Соломона . . . 47
1.6. Полиномиальные и мономиальные коды . . . 47
1.7. Выводы . . . 48

Глава 2. Многочленные коды . . . 50
2.1. Некоторые обобщения . . . 50
2.1.1. Многочленные коды . . . 50
2.1.2. Обобщенные каскадные коды с перекрестными связями . . . 52
2.1.3. Обобщенное разложение Плоткина . . . 53
2.2. Динамически замороженные символы . . . 54
2.3. Анализ вероятности ошибки декодирования . . . 58
2.4. Расширенные коды БЧХ . . . 59
2.4.1. Структура ограничений динамического замораживания . . . 59
2.4.2. Численные результаты . . . 62
2.5. Расширенный код Голея . . . 64
2.6. Расширенные коды Рида-Соломона . . . 66
2.6.1. Структура ограничений динамического замораживания . . . 66
2.6.2. Метод последовательного исключения . . . 67
2.7. Выводы . . . 69

Глава 3. Полярные подкоды . . . 71
3.1. Расчет надежностей подканалов в поляризующем преобразовании Арикана . . . 71
3.1.1. Аддитивный гауссовский канал . . . 71
3.1.1.1. Метод Гауссовской аппроксимации . . . 71

3.1.1.2. Численные результаты . . . 73
3.1.2. Релеевский канал . . . 75
3.1.2.1. Двумерный метод . . . 76
3.1.2.2. Упрощенный одномерный метод . . . 77
3.1.2.3. Численные результаты . . . 80
3.2. Полярные подкоды . . . 81
3.2.1. Полярные подкоды с одним поляризующим преобразованием . . . 83
3.2.1.1. Полярные подкоды кодов РБЧХ . . . 83
3.2.1.2. Рандомизированные полярные подкоды . . . 90
3.2.2. Цепные полярные подкоды . . . 102
3.2.2.1. Методы комбинирования кодов . . . 103
3.2.2.2. Алгебраическая конструкция . . . 103
3.2.2.3. Рандомизированная конструкция . . . 108
3.2.3. Полярные подкоды для систем мобильной связи 5 поколения . . . 115
3.2.4. Звездные полярные подкоды . . . 117
3.2.4.1. Общий случай . . . 118
3.2.4.2. Симметричные звездные полярные подкоды . . . 119
3.2.4.3. Рандомизированная конструкция на основе кодов РБЧХ . . . 121
3.2.4.4. Численные результаты . . . 122
3.3. Выводы . . . 124

Глава 4. Декодирование многочленных кодов . . . 125
4.1. Последовательный алгоритм декодирования . . . 125
4.1.1. Общее описание . . . 125
4.1.2. Весовая функция . . . 128
4.1.2.1. Эффективное вычисление весовой функции . . . 131
4.1.2.2. Эвристическая функция . . . 133
4.1.2.3. Обсуждение . . . 134
4.1.2.4. Сводка предлагаемого подхода . . . 134
4.1.3. Блочный последовательный алгоритм . . . 135
4.1.4. Гибридное декодирование . . . 136
4.1.5. Численные результаты . . . 139
4.2. Декодирование цепных полярных подкодов . . . 144
4.2.1. Обобщенный алгоритм последовательного исключения . . . 144

4.2.2. Списочное декодирование . . . 146
4.2.3. Жадное расписание . . . 147
4.2.4. Численные результаты . . . 148
4.3. Декодирование расширенного кода Голея . . . 149
4.3.1. Списочное и последовательное декодирование . . . 149
4.3.2. Блочное декодирование . . . 150
4.4. Параллельное списочное декодирование звездных полярных подкодов . . . 152
4.5. Декодирование кодов Рида-Соломона . . . 155
4.5.1. Последовательное декодирование . . . 155
4.5.2. Перестановочное декодирование . . . 155
4.5.3. Численные результаты . . . 156
4.6. Выводы . . . 158

Глава 5. Списочное декодирование кодов Рида-Соломона . . . 160
5.1. Быстрая интерполяция в алгоритме Гурусвами-Судана . . . 160
5.1.1. Базис модуля интерполяционных многочленов . . . 160
5.1.1.1. Свойства . . . 160
5.1.1.2. Преобразование базиса модуля . . . 162
5.1.2. Двоичный интерполяционный алгоритм . . . 163
5.1.2.1. Интерполяция посредством перемножения идеалов . . . 163
5.1.2.2. Анализ сложности . . . 169
5.1.2.3. Перекодирование . . . 173
5.1.3. Численные результаты . . . 175
5.2. Быстрая интерполяция в алгоритме Ву . . . 177
5.2.1. Описание алгоритма Ву в частотной области . . . 177
5.2.2. Интерполяция частично однородными многочленами . . . 179
5.2.3. Эффективная интерполяция . . . 182
5.2.3.1. Частично однородные многочлены . . . 182
5.2.3.2. Расширение модуля . . . 185
5.2.3.3. Наращивание кратности корней . . . 188
5.2.3.4. Двоичный интерполяционный алгоритм . . . 190
5.2.4. Численные результаты . . . 190
5.3. Выводы . . . 192

Глава 6. Спектральные методы кодирования и декодирования кодов Рида-Соломона . . . 193
6.1. Быстрое преобразование Фурье над конечным полем . . . 193
6.1.1. Циклотомический алгоритм БПФ . . . 193
6.1.1.1. Циклотомическое разложение . . . 194
6.1.1.2. Быстрое вычисление преобразования Фурье . . . 194
6.1.1.3. Оценка сложности алгоритма . . . 197
6.1.1.4. Сравнение сложности алгоритмов быстрого преобразования Фурье . . . 197
6.1.2. Обратный циклотомический алгоритм . . . 198
6.2. Применение обратного преобразования Фурье для быстрого вычисления вектора синдрома . . . 201
6.3. Быстрое систематическое кодирование кодов Рида-Соломона . . . 204
6.3.1. Кодирование через исправление стираний . . . 204
6.3.2. Эффективное вычисление синдрома . . . 206
6.3.2.1. Вычисление синдрома информационных символов . . . 207
6.3.2.2. Вычисление синдрома проверочных символов . . . 209
6.3.3. Выбор проверочных символов . . . 210
6.4. Применение кодов Рида-Соломона в системах хранения данных . . . 212
6.4.1. Компромисс между скоростью кодирования и восстановления . . . 212
6.4.2. Частичное обновление . . . 214
6.4.3. Сводное описание предлагаемого подхода . . . 214
6.4.4. Численные результаты . . . 215
6.5. Быстрое систематическое кодирование полярных кодов с ядром Рида-Соломона 218
6.5.1. Декодирование полярных кодов с исправлением стираний . . . 218
6.5.1.1. Итеративный алгоритм исправления стираний . . . 219
6.5.1.2. Эффективная реализация шага А . . . 221
6.5.2. Быстрое систематическое кодирование . . . 223
6.5.2.1. Расположение проверочных символов . . . 223
6.5.2.2. Эффективная реализация шага Б . . . 226
6.6. Ускоренный поиск корней многочленов над конечными полями . . . 228
6.6.1. Аффинное разложение . . . 228
6.6.2. Специальные разложения . . . 230
6.6.3. Обобщенное разложение . . . 231

6.6.4. Гибридный алгоритм поиска корней многочленов 232
6.7. Выводы 233

Заключение 235

Список литературы 238

Список сокращений и условных обозначений 253

Введение

Актуальность темы исследования. Быстрый рост объема цифровых данных, производимых и обрабатываемых различными техническими системами, требует создания соответствующей инфраструктуры для их передачи и хранения. При этом постоянно ужесточаются требования к скорости обмена данными, задержке передачи информации, энергетической и спектральной эффективности, стоимости оборудования и т. п. Их выполнение невозможно без использования методов помехоустойчивого кодирования.

Несмотря на долгую историю развития теории кодирования, построение корректирующих кодов и методов их декодирования, которые удовлетворяли бы требованиям к перспективным техническим системам, вызывает серьезные затруднения. В частности, современные информационные системы зачастую используют обмен короткими сообщениями, которые должны доставляться с низкой задержкой. Однако оказывается, что широко используемые в настоящее время коды с циклическим замыканием, турбо-коды и коды с малой плотностью проверок на четность (LDPC) требуют внесения весьма высокой избыточности для обеспечения приемлемой вероятности ошибки декодирования таких сообщений. Кроме того, задержка обработки информации в декодере оказывается неудовлетворительной, а высокое энергопотребление приемо-передающей аппаратуры, зависящее в значительной степени от сложности используемых алгоритмов кодирования и декодирования, существенно ограничивает время автономной работы соответствующих устройств.

Причиной этого является отсутствие простых алгоритмов декодирования (почти) по максимуму правдоподобия для известных хороших кодов (например, БЧХ, Рида-Соломона), и недостаточная корректирующая способность тех кодов (например, кодов с малой плотностью проверок на четность), для которых имеются эффективные алгоритмы мягкого декодирования. В частности, предложенные в 2008 г. Э. Ариканом полярные коды достигают предела Шеннона, и для них известны простые алгоритмы построения, кодирования и декодирования. Однако на практике оказывается, что корректирующая способность полярных кодов с практически значимыми длинами (до нескольких тысяч) существенно хуже по сравнению с другими известными кодами, алгоритм последовательного исключения не обеспечивает декодирование полярных кодов по максимуму правдоподобия, сложность (размер списка) усовершенствованного списочного алгоритма последовательного исключения чрезмерно высока, а алгоритм Тала-Варди эволюции плотностей требует весьма большого числа уровней квантования, что также приводит к слишком большой сложности построения кодов. Простая структура и замечательные асимптотические свойства полярных кодов делают их

весьма привлекательными для использования в перспективных системах передачи информации. Однако для удовлетворения требованиям, предъявляемым к таким системам, должна быть решена задача построения упрощенных алгоритмов декодирования полярных кодов и усовершенствования их конструкции с целью повышения корректирующей способности.

Многие современные системы хранения и передачи информации используют коды Рида-Соломона. Известные алгоритмы с полиномиальной сложностью не обеспечивают их декодирование по максимуму правдоподобия. Одним из возможных путей повышения практической корректирующей способности кодов Рида-Соломона является использование списочного декодирования. Однако алгоритмы Гурусвами-Судана и Ву списочного декодирования имеют весьма высокую (хотя и полиномиальную) сложность, что ограничивает их практическое применение. В связи с этим, для повышения помехозащищенности современных и перспективных систем хранения и передачи информации должны быть разработаны простые алгоритмы декодирования кодов Рида-Соломона с улучшенной корректирующей способностью, а также снижена сложность известных списочных алгоритмов декодирования.

В системах хранения данных применение длинных МДР-кодов, обеспечивающих оптимальную избыточность при заданной вероятности потери информации, сдерживается высокой сложностью их кодирования и значительным объемом данных, которые приходится считывать и/или передавать в процессе восстановления. Хотя недавно были предложены эффективные решения последней проблемы для кодов Рида-Соломона, недостаточная производительность операции кодирования, регулярно применяемой в штатном режиме работы СХД, заставляет разработчиков применять субоптимальные с точки зрения избыточности решения. Таким образом, для повышения производительности и полезной емкости систем хранения данных должна быть решена задача построения упрощенных алгоритмов систематического кодирования кодов Рида-Соломона.

Коды Рида-Соломона, Рида-Маллера, полярные, БЧХ используют весьма схожую конструкцию. Более конкретно, кодовые слова этих кодов получаются как векторы значений некоторых (различным образом определенных) многочленов в различных точках. Вместе с тем, формально эти коды друг к другу не сводятся. В связи с этим возникает идея создания единого математического аппарата для описания указанных классов кодов. Это позволило бы не только упростить их декодирование за счет расширения области применимости известных эффективных алгоритмов, но и построить новые коды, которые сочетали бы в себе хорошую корректирующую способность и возможность простого декодирования. Кроме того, необходимо разработать такую кодовую конструкцию, которая позволяла бы выбирать корректирующую способность с учетом ограничений на сложность декодирования. Решению

этих проблем посвящена данная работа.

Значительный вклад в решение проблем построения кодов с высокой корректирующей способностью, создания эффективных методов их декодирования и исследования их свойств внесли в нашей стране В.Б. Афанасьев, А.М. Барг, Л.А. Бассылыго, Э.Л. Блох, С.В. Беззатеев, И.Е. Бочарова, Э.М. Габидулин, И.И. Думер, С.И. Егоров, К. Ш. Зигангиров, В.А. Зиновьев, В.В. Зяблов, Г.А. Кабатянский, В.Д. Колесник, Е.А. Крук, Б.Д. Кудряшов, Е.Т. Мирончиков, С.В. Федоренко, Н.А. Шехунова, за рубежом Э. Арикан, Э. Берлекэмп, Р. Блейхут, А. Варди, Й. Ву, Р. Галлагер, В. Гурусвами, Т. Касами, Р. Кётер, С. Кудекар, С. Кумар, Дж. Месси, Р. Рот, М. Судан, И. Тал, Т. Танака, Р. Урбанке, М. Фоссорье и многие другие.

Цели и задачи диссертационной работы. Целью работы является разработка математического аппарата, который позволил бы упростить декодирование корректирующих кодов, а также построить новые коды, обеспечивающие повышение помехозащищенности и производительности систем передачи и хранения информации. При этом *объектом исследований* являются линейные блоковые коды, а *предметом исследований* — методы их построения и декодирования.

Для достижения указанных целей в работе решаются следующие задачи:

1. Разработка математической модели линейных блоковых кодов, основанной на представлении их кодовых слов как векторов значений (векторных) многочленов от нескольких переменных в различных точках, позволяющей использовать для декодирования метод последовательного исключения и его обобщения.

2. Разработка на основе созданной модели метода построения кодов, называемых в работе полярными подкодами, который допускал бы возможность выбора корректирующей способности с учетом ограничений на сложность декодирования.

3. Применение предложенной модели для построения новых алгоритмов декодирования полярных кодов, кодов БЧХ, Рида-Соломона, Голея и полярных подкодов.

4. Разработка быстрых алгоритмов кодирования и декодирования кодов Рида-Соломона.

5. Разработка на основе предложенных подходов методов помехозащиты для перспективных систем передачи и хранения информации.

Научная новизна. В работе введено обобщение полиномиальных и мономиальных кодов, называемое многочленными кодами. Оно основано на представлении линейного блокового кода в виде системы линейных ограничений динамического замораживания (ОДЗ) на

входные символы поляризующего преобразования. С помощью предложенного обобщения показана возможность использования метода последовательного исключения и его аналогов для декодирования кодов БЧХ, Голея и Рида-Соломона. Впервые представлена конструкция полярных подкодов, основанная на предложенном обобщении и обеспечивающая существенно лучшую корректирующую способность по сравнению с полярными кодами с CRC и известными кодами с малой плотностью проверок на четность. Впервые предложен последовательный алгоритм декодирования полярных (под)кодов. Впервые предложен рандомизированный алгоритм построения базиса Грёбнера произведения нульмерных идеалов и описано применение двоичного метода возведения в степень к задаче построения базиса Грёбнера идеала/модуля интерполяционных многочленов в алгоритмах Гурусвами-Судана и Ву списочного декодирования кодов Рида-Соломона. Впервые представлен асимптотически быстрый вариант циклотомического алгоритма быстрого преобразования Фурье и основанный на нем быстрый алгоритм систематического кодирования кодов Рида-Соломона.

Все полученные результаты на момент их публикации являлись новыми.

Теоретическая и практическая значимость. Предложенный в работе метод представления линейных блоковых кодов в виде системы ограничений динамического замораживания может быть использован для создания новых методов декодирования корректирующих кодов, а также для создания новых корректирующих кодов. Представленный быстрый алгоритм построения базиса Грёбнера произведения нульмерных идеалов может быть использован в системах компьютерной алгебры.

Представленные в работе быстрые алгоритмы декодирования кодов БЧХ и Рида-Соломона могут быть использованы в существующих и перспективных системах хранения и передачи информации, в которых применяются соответствующие коды. Предложенные в работе полярные подкоды могут быть использованы в перспективных системах мобильной и фиксированной связи. В частности, предложенная конструкция рандомизированных полярных подкодов положена в основу процедуры кодирования для контрольного канала, предусмотренной в стандарте мобильной связи 5 поколения [182, п. 5.3.1.2], и продемонстрировала лучшую корректирующую способность по сравнению с кодами с циклическим замыканием, применяемыми в системах предшествующих поколений. Предложенный быстрый алгоритм систематического кодирования кодов Рида-Соломона может быть использован в системах хранения данных.

Положения, выносимые на защиту:

1. Представление линейных блоковых кодов в виде системы ограничений динамического

замораживания (ОДЗ) на элементы входного вектора поляризующего преобразования позволяет осуществлять декодирование некоторых линейных блоковых кодов (в частности, БЧХ, Голея и Рида-Соломона) с использованием алгоритма последовательного исключения и его списочного/последовательного обобщения.

2. Метод построения полярных подкодов, основанный на построении системы ОДЗ, обеспечивающей исключение ненулевых кодовых слов малого веса, и наложении ограничений статического замораживания на символы, передаваемые по ненадежным подканалам поляризующего преобразования, позволяет получить коды с улучшенной корректирующей способностью по сравнению с известными LDPC и турбо-кодами.

3. Последовательный алгоритм обеспечивает снижение средней сложности декодирования полярных подкодов по сравнению со списочным и стековым алгоритмами с аналогичной корректирующей способностью.

4. Быстрый алгоритм двумерной интерполяции, основанный на рандомизированном алгоритме построения базиса Грёбнера произведения нульмерных идеалов и двоичном методе возведения в степень, обеспечивает снижение сложности алгоритмов Гурусвами-Судана и Ву списочного декодирования кодов Рида-Соломона.

5. Циклотомический алгоритм БПФ обеспечивает быстрое систематическое кодирование кодов Рида-Соломона и повышение производительности операции кодирования в системах хранения данных.

Степень достоверности и апробация результатов. Основные результаты диссертации докладывались на следующих конференциях:

1. IEEE International Symposium on Information Theory (2004,2014 ,2017);

2. IEEE Information Theory Workshop (2012,2013);

3. Int. Symposium on Information Theory and its Applications (2014);

4. Polar Coding for Future Networks: Theory and Practice, Polar Coding in Wireless Communications: Theory and Implementation (2017,2018);

5. Int. ITG Conference on Systems, Communications and Coding (2017);

6. Int. Symposium on turbo codes and iterative information processing (2016);

7. International Symposium on Wireless Communication Systems (2015).

8. Iran Workshop on Communication and Information Theory (2018).

Кроме того, они были представлены на следующих семинарах:

1. по теории кодирования Института проблем передачи информации РАН (Москва, руководитель — Л.А. Бассалыго);

2. по теории информации и ее приложениям Калифорнийского университета в Сан-Диего (США, руководитель — А. Варди);

3. лаборатории связи и построения сигналов Пхохангского университета (Южная Корея, руководитель — К. Янг);

4. кафедры безопасности информационных систем Санкт-Петербургского государственного университета аэрокосмического приборостроения (руководитель — Е. А. Крук);

5. кафедры распределенных вычислений и компьютерных сетей Санкт-Петербургского политехнического университета (руководитель — Ю.Г. Карпов).

Все предложенные алгоритмы были реализованы программно, их поведение было исследовано методами статистического моделирования, результаты которого были сопоставлены с ранее опубликованными.

Публикации. Материалы диссертации опубликованы в 26 печатных работах, из них 12 статей в рецензируемых журналах [64, 23, 48, 21, 165, 175, 166, 22, 119, 170, 178, 61], 14 статей в сборниках трудов конференций [112, 164, 167, 176, 168, 177, 169, 180, 171, 172, 91, 179, 173, 174].

Личный вклад автора. Содержание диссертации и основные положения, выносимые на защиту, отражают персональный вклад автора в опубликованные работы. Некоторые результаты получены совместно с соавторами (С.В. Федоренко, В.Д. Милославской, Г.А. Трофимюком, К.Г. Ивановым, Е. Коста, А. Варди, Ю. Ма, М.Х. Ли), которым автор выражает искреннюю благодарность. В работе [64] автору принадлежит метод разложения многочленов на сумму полиномов, кратных аффинным. В работе [23] автору принадлежит идея представления полинома в виде суммы линеаризованных многочленов и разложения матрицы дискретного преобразования Фурье на двоичную и блочно-диагональную матрицы. В работе [112] автору принадлежит матричная интерпретация итеративного интерполяционного алгоритма, в то время как доказательство теоремы 3 было получено соавторами независимо друг от друга. В работе [48] автору принадлежит конструкция обратного циклотомического алгоритма БПФ, а также идея варьирования порядка элементов в нормальном базисе с

целью снижения сложности получаемого алгоритма. В работе [175] автору принадлежит новая интерпретация алгоритма Ву, оценка размера получаемого списка, а также обобщение двоичного интерполяционного алгоритма. В работе [176] автору принадлежит идея использования кодов БЧХ для построения ограничений динамического замораживания, а также идея использования эвристической функции для ускорения декодирования полярных кодов. В работе [119] автору принадлежит идея использования эвристической функции для ускорения декодирования полярных кодов. В работах [180, 91] автору принадлежит идея использования обобщенного разложения Плоткина для декодирования рассматриваемых кодов. В работе [178] автору принадлежит идея использования кодов БЧХ для построения ограничений динамического замораживания, теоремы 1 и 2, а также конструкции полярных подкодов с ядрами БЧХ и Рида-Соломона. В работе [179] автору принадлежит конструкция динамически замороженных символов типа Б и идея использования псевдослучайных ограничений динамического замораживания.

Автор выражает благодарность Г.А. Трофимюку, Н.В. Якубе, С.П. Рецу, К.Г. Иванову и Р.А. Морозову за помощь в программной реализации и экспериментальном исследовании некоторых алгоритмов.

Структура и объем диссертации. Диссертация состоит из введения, шести глав, заключения и библиографии. Общий объем диссертации 255 страниц, включая 63 рисунка и 12 таблиц. Библиография включает 196 наименований.

В главе 1 представлены основные понятия и определения, обзор актуальных технических требований к средствам помехоустойчивого кодирования, некоторых кодовых конструкций и алгоритмов их декодирования. Рассматриваются конструкции обобщенных каскадных и полярных кодов, многошаговый метод декодирования и алгоритм последовательного исключения, анализируются причины их недостаточной корректирующей способности. Рассматриваются методы кодирования и декодирования кодов Рида-Соломона. Показано, что наиболее сложным этапом списочного декодирования кодов Рида-Соломона является интерполяционный шаг. На основе проведенного анализа технических требований и существующих подходов к построению и декодированию обобщенных каскадных и полярных кодов, кодов Рида-Соломона сформулированы основные задачи, решаемые в работе.

В главе 2 введено обобщение известных классов полиномиальных и мономиальных кодов, называемое многочленными кодами, а также связанные с ним понятия ограничений динамического замораживания (ОДЗ) на входные символы поляризующего преобразования, обобщенных каскадных кодов с перекрестными связями и обобщенного разложения Плоткина. Показано, что представление линейных блоковых кодов в виде системы ОДЗ позволяет

использовать для их декодирования метод последовательного исключения и его обобщения. Охарактеризована структура ограничений динамического замораживания для расширенных примитивных кодов БЧХ (РБЧХ) и Рида-Соломона в узком смысле. Показано, что использование последовательного алгоритма декодирования, введенного в разделе 4.1, позволяет выполнить декодирование коротких кодов РБЧХ со сложностью меньшей, чем в случае известного алгоритма BEAST. В случае расширенного кода Голея применение предложенного подхода позволяет выполнить декодирование по максимуму правдоподобия со средней сложностью, сопоставимой со сложностью алгоритма Варди, являющегося наилучшим известным методом декодирования этого кода. Дальнейшее развитие этого подхода в случае кодов Рида-Соломона рассмотрено в разделе 4.5, где свойство инвариантности множества номеров замороженных символов относительно некоторого класса перестановок символов кодового слова используется для построения алгоритма декодирования коротких кодов Рида-Соломона с корректирующей способностью лучшей, чем у известного адаптивного алгоритма распространения доверия.

В главе 3 представлены новые конструкции многочленных кодов, допускающих быстрое декодирование с хорошей корректирующей способностью, основанные на введенном в главе 2 обобщении. Предложен класс полярных подкодов, а также два подхода к их построению. Оба подхода предполагают построение некоторой системы ОДЗ. Первый подход предполагает использование некоторого протокода с достаточно большим минимальным расстоянием. Предложен метод нахождения такого подкода заданного протокода, который минимизировал бы вероятность ошибки декодирования методом последовательного исключения. Описаны процедуры построения полярных подкодов с ядрами Арикана, РБЧХ и Рида-Соломона. Второй подход предполагает построение подкода классического полярного кода, из которого исключаются ненулевые кодовые слова малого веса. Введены два типа ограничений динамического замораживания. Ограничения типа А обеспечивают минимизацию ошибочного коэффициента получаемого кода, в то время как ограничения типа Б позволяют снизить вероятность потери списочным или последовательным декодером правильного пути на ранних фазах декодирования. Представлены методы построения цепных полярных подкодов, которые позволяют получить произвольную длину кода без использования выкалывания или укорочения, что позволяет упростить реализацию декодера. Предложена конструкция звездных полярных подкодов, которая допускает декодирование с сокращенной задержкой по сравнению с классическим методом последовательного исключения и его аналогами. Кроме того, представлены упрощенные методы расчета надежности подканалов поляризующего преобразования Арикана для случая каналов с аддитивным белым гауссовским шумом

и независимыми релеевскими замираниями. Приведены результаты статистического моделирования, свидетельствующие о том, что предлагаемые коды обеспечивают существенно лучшую корректирующую способность по сравнению с известными турбо- и LDPC кодами, кодами с циклическим замыканием и полярными кодами с CRC. Кроме того, показано, что, варьируя параметры предложенной кодовой конструкции и представленных в главе 4 алгоритмов декодирования, можно выбирать компромисс между сложностью декодирования и корректирующей способностью.

В главе 4 представлены новые методы декодирования полярных подкодов. Предложен последовательный алгоритм декодирования и функция веса пути для него. Применение представления линейных блоковых кодов, основанного на предложенном в главе 2 обобщении, позволяет использовать предложенный алгоритм также для декодирования расширенного кода Голея, коротких кодов БЧХ и Рида-Соломона. Показано, что применение последовательного алгоритма к предложенным полярным подкодам позволяет получить меньшую сложность декодирования при лучшей корректирующей способности по сравнению с LDPC кодами, описанными в стандарте CCSDS. Показано, что дальнейшее снижение сложности декодирования может быть достигнуто за счет рекурсивного использования обобщенного разложения Плоткина и стандартных методов декодирования коротких кодов Рида-Маллера. Предложено обобщение списочного алгоритма Тала-Варди и последовательного алгоритма на случай цепных полярных подкодов. Предложен метод параллельного декодирования звездных полярных подкодов. Показано, что применение последовательного алгоритма в сочетании с методами перестановочного декодирования позволяет выполнить декодирование коротких кодов Рида-Соломона с существенно лучшей корректирующей способностью по сравнению с адаптивным алгоритмом распространения доверия в сочетании с мягким алгебраическим декодированием, наилучшим известным в настоящее время методом декодирования кодов Рида-Соломона с полиномиальной сложностью.

В главе 5 представлены методы снижения сложности списочного декодирования длинных кодов Рида-Соломона. Предложено использование двоичного метода возведения в степень для построения идеала интерполяционных многочленов в алгоритме Гурусвами-Судана. Предложен рандомизированный алгоритм построения базиса Грёбнера произведения нульмерных идеалов многочленов. Представлено обобщение этого подхода на случай алгоритма Ву. Получена уточненная оценка на размер списка, получаемого в алгоритме Ву. Показано, что применение предложенных подходов обеспечивает снижение асимптотической сложности интерполяционного шага в алгоритмах Гурусвами-Судана и Ву по сравнению с итеративным интерполяционным алгоритмом и алгоритмом Ли-О'Салливана, а при программной реали-

зации — многократное повышение производительности списочного декодера.

В главе 6 представлен циклотомический алгоритм быстрого вычисления дискретного преобразования Фурье над конечным полем и основанный на нем быстрый алгоритм систематического кодирования кодов Рида-Соломона. Представлен новый метод построения последовательности операций, реализующей быстрое умножение двоичной матрицы на вектор. Показано, что при программной реализации применение предложенного подхода обеспечивает повышение производительности операции систематического кодирования (до 3.6 раз) по сравнению с библиотекой Jerasure, широко используемой в современных системах хранения данных. Представлено обобщение предложенного подхода на случай систематического кодирования полярных кодов с ядром Рида-Соломона. Циклотомический алгоритм БПФ основан на представлении многочлена над конечным полем в виде суммы линеаризованных многочленов. Показано, что применение аналогичного подхода позволяет также упростить операцию поиска корней многочленов над конечным полем, которая используется в классических и списочных алгоритмах декодирования кодов Рида-Соломона.

Глава 1

Корректирующие коды и методы их декодирования

В данной главе представлен обзор некоторых проблем, связанных с использованием методов помехоустойчивого кодирования в современных системах передачи и хранения информации.

1.1. Технические требования к средствам помехоустойчивого кодирования

Помехоустойчивое кодирование является неотъемлемой частью современных и перспективных систем передачи и хранения информации. Несмотря на долгую историю развития теории кодирования, построение корректирующих кодов и методов их декодирования, которые обеспечивали бы выполнение постоянно ужесточающихся технических требований, встречает серьезные затруднения. Различные приложения требуют построения корректирующих кодов с различными соотношениями скорости, минимального расстояния, вероятности ошибки декодирования, сложности и задержки кодирования и декодирования и других параметров.

1.1.1. Системы передачи информации

Одним из ключевых требований к сетям мобильной связи 5 поколения является обеспечение пиковой скорости передачи данных до 50 Гб/с (в 50 раз больше, чем в стандарте LTE) с задержкой не более 1 мс (в 10 раз меньше, чем в стандарте LTE) [69, 53]. Это требует не только улучшения спектральной эффективности используемых сигнально-кодовых конструкций, но и существенного повышения производительности кодеров и декодеров используемых корректирующих кодов. Ввиду большой распространенности мобильных устройств, при этом также требуется значительно снизить энергопотребление кодеров и декодеров, которое определяется сложностью используемых алгоритмов, а также снизить мощность сигнала (т.е. отношение сигнал/шум), требуемую для достижения заданной вероятности ошибки. Кроме того, такие системы должны обеспечивать эффективную поддержку межмашинной коммуникации, что требует обеспечения хорошей помехозащиты коротких сообщений.

1.1.2. Системы хранения данных

Быстрый рост объема данных, доступных в цифровой форме, требует создания соответствующих систем их хранения. Такие системы включают в себя большое число (до нескольких десятков и сотен тысяч) отдельных устройств хранения данных (например, жестких дисков), которые подвержены отказам. Для защиты данных необходимо использовать методы помехоустойчивого кодирования. В больших современных системах хранения данных отказы (постоянные или временные) устройств являются обычным режимом работы. В этих условиях для обеспечения постоянной доступности данных приходится использовать корректирующие коды с достаточно большим минимальным расстоянием. Например, в системах Hadoop File System и EMC Atmos предусмотрено использование кодов Рида-Соломона, соответственно, с 4 и 6 проверочными символами. Производительность такого рода систем определяется:

1. Сложностью операции кодирования, которая выполняется при записи данных.
2. Числом операций (в т.ч. ввода/вывода), выполняемых при частичном обновлении данных.
3. Числом устройств, с которых производится считывание данных в процессе восстановления информации, хранившейся на отказавших устройствах.
4. Объемом данных, передаваемых по сети (в случае сетевых систем хранения) в процессе восстановления информации, хранившейся на отказавших устройствах.
5. Эффективностью используемых механизмов кеширования, упреждающего чтения, размещения данных по носителям различных типов и др.

За последние годы опубликовано большое число работ, посвященных построению локально восстановимых и регенерирующих кодов (см., например, [134, 128]), использование которых позволяет существенно повысить эффективность операции восстановления данных. Однако в литературе уделено недостаточно внимания производительности операции кодирования, хотя она регулярно выполняется в штатном режиме работы систем хранения.

1.2. Основные понятия теории кодирования

В данном разделе представлены основные понятия теории информации и кодирования, используемые далее в работе.

1.2.1. Теоретико-информационные основы

Для обеспечения надежной передачи информации по зашумленным каналам требуется использовать помехоустойчивое кодирование, т.е. некоторый способ внесения избыточности в передаваемые данные. Блоковым кодом длины n над некоторым алфавитом $\mathcal{X}, |\mathcal{X}| = q$, называется некоторое множество $\mathcal{C} \subset \mathcal{X}^n$ векторов мощности M. Кодирование блоковым кодом $\mathcal{C}$ предполагает, что данные разбиваются на блоки по $\log_q M$ q-ичных символов, которые заменяются некоторым определенным образом на элементы кода (кодовые слова). Скоростью кода называется величина $R = \frac{\log_q M}{n}$. В данной работе будем считать, что $\mathcal{X}$ является конечным полем $\mathbb{F}_q$.

Ограничимся рассмотрением симметричных стационарных каналов без памяти с дискретным входом, т.е. таких каналов, для которых вероятность $W\left\{y_j^{j+n-1} | x_j^{j+n-1}\right\}$ приема вектора $y_j^{j+n-1} = (y_j, \ldots, y_{j+n-1})$ при условии передачи x_j^{j+n-1}, где $a_u^v = (a_u, a_{u+1}, \ldots, a_v)$, определяется исключительно значениями x_i и y_i, причем

$$W\left\{y_j^{j+n-1} | x_j^{j+n-1}\right\} = \prod_{i=j}^{j+n-1} W_{Y|X}\left\{y_i | x_i\right\}.$$

Здесь $y_i \in \mathcal{Y}, x_i \in \mathcal{X}$, где $\mathcal{Y}$ — выходной алфавит канала, X и Y — случайные величины, соответствующие его входу и выходу, и $W_{Y|X}\left\{y_i | x_i\right\}$ — функция переходных вероятностей канала. Существуют различные подходы к определению симметричных каналов. В данной работе будем считать [126], что канал является симметричным, если для каждого $x \in \mathbb{F}_q$ существует такая перестановка $\sigma_x : \mathcal{Y} \to \mathcal{Y}$, что

$$\forall y \in \mathcal{Y}, \forall x, x' \in \mathbb{F}_q : W_{Y|X}\left\{y | x\right\} = W_{Y|X}\left\{\sigma_{x'-x}(y) | x'\right\}.$$

Это соответствует каналу симметричному по входу в смысле [10] и симметричному по выходу в смысле [146].

На стороне приемника целесообразно использовать прием по максимуму апостериорной вероятности, т.е. принимать решение в пользу кодового слова (и соответствующего сообщения)

$$\hat{c} = \arg\max_{c \in \mathcal{C}} W_{X^n|Y^n}\left\{c | y\right\},$$

где $P\left\{c\right\}$ — вероятность выбора передатчиком кодового слова c.

В случае равновероятных сообщений и канала без памяти этот критерий эквивалентен критерию максимума правдоподобия

$$\hat{c} = \arg\max_{c \in \mathcal{C}} \prod_{i=0}^{n-1} W_{Y|X}\left\{y_i | c_i\right\}.$$

К. Шенноном было показано, что для любого дискретного стационарного канала с пропускной способностью C для любого $\delta > 0$ существует такое $\epsilon > 0$, что при использовании любого кода со скоростью $R > C + \delta$ средняя вероятность ошибки удовлетворяет $P_e \geq \epsilon$. С другой стороны, для любого дискретного стационарного канала без памяти с пропускной способностью C для любых $\epsilon, \delta > 0$ существует такое $n_0 \in \mathbb{N}$, что для всех $n \geq n_0$ существует код длиной n со скоростью $R \geq C - \delta$, средняя вероятность ошибки которого $P_e \leq \epsilon$. Пропускная способность канала равна

$$C = \max_{W_X\{x\}} I(X;Y),$$

где максимизация производится по распределению символов на его входе, и

$$I(X;Y) = \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} W_{XY}\{x,y\} \log \frac{W_{Y|X}\{y|x\}}{W_Y\{y\}} \tag{1.1}$$

— средняя взаимная информация случайных величин X, Y. Здесь $W_Y\{y\}$ — распределение символов на выходе канала, на вход которого подаются случайные величины, распределенные по закону $W_X\{x\}$. Для каналов, рассматриваемых в данной работе, оптимальным распределением является равномерное. В случае каналов с непрерывным выходом вероятности $W_{Y|X}\{y|x\}$ в вышеприведенных выражениях должны быть заменены на плотности распределения $W_{Y|X}(y|x)$, а суммы по y — на интегралы.

1.2.2. Линейные блоковые коды

Из соображений простоты реализации целесообразно ограничиться использованием линейных блоковых кодов. (n,k,d) линейным блоковым кодом $\mathcal{C}$ над $\mathbb{F}_q$ называется k-мерное линейное подпространство векторного пространства $\mathbb{F}_q^n$, все элементы которого различаются не менее чем в d позициях, т.е.

$$\mathcal{C} = \left\{c = xG | x \in \mathbb{F}_q^k\right\} = \left\{c \in \mathbb{F}_q^n \,|\, Hc^T = 0\right\},$$

где G — $k \times n$ порождающая матрица, строки которой образуют базис $\mathcal{C}$ как линейного пространства, H — $(n-k) \times n$ проверочная матрица полного ранга, удовлетворяющая $GH^T = 0$, n — длина кода, k — его размерность, и d — минимальное расстояние. В тех случаях, когда оно неизвестно или не имеет значения, будем обозначать параметры кода как (n,k).

Можно показать, что для любого симметричного канала без памяти существуют линейные коды, обеспечивающие сколь угодно надежную передачу данных со скоростью $R < C$. Примерами таких кодов являются каскадные коды [68], коды на графах-расширителях [36] и пространственно связанные коды [103]. Недавно было показано, что коды Рида-Маллера

и расширенные примитивные коды БЧХ в узком смысле достигают пропускной способности двоичного стирающего канала [145].

1.2.3. Вероятность ошибки декодирования

Можно показать, что для канала без памяти с дискретным входом для любого натурального n и $R > 0$ ансамбль блоковых кодов длины n со скоростью R, в котором символы всех кодовых слов выбираются независимо с распределением $W_X\{x\}$, обеспечивает среднюю вероятность ошибки декодирования по максимуму правдоподобия

$$P_e \leq 2^{-nE_r(R)},$$

где $E_r(R)$ — экспонента случайного кодирования, которая равна [10]

$$E_r(R) = \max_{0\leq\rho\leq 1}(E_0(\rho) - \rho R), \tag{1.2}$$

где

$$E_0(\rho) = -\log_2\left(\int_{\mathcal{Y}}\left(\sum_{x\in\mathcal{X}} W_X\{x\}\left(W_{Y|X}(y|x)\right)^{\frac{1}{1+\rho}}\right)^{1+\rho} dy\right). \tag{1.3}$$

На практике с целью снижения задержки и сложности декодирования желательно минимизировать длину используемых кодов. В связи с этим представляет интерес нахождение минимальной возможной длины n_0 кода, который был бы способен обеспечить заданную вероятность ошибки при заданной скорости $R < C$. В работе [141] было показано, что для случайных кодов справедлива оценка

$$n_0 \approx \left(\frac{Q^{-1}(P_e)}{C-R}\right)^2 V,$$

где $Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt$ и V — параметр, называемый канальной дисперсией. Таким образом, минимальная потребная длина кода ведет себя как $n_0 = O(\frac{1}{(C-R)^\mu})$, где $\mu = 2$ — экспонента масштабирования. Представляет большой практический интерес построение конкретных семейств корректирующих кодов, которые обладали бы эффективными алгоритмами декодирования и имели бы экспоненту масштабирования, близкую к 2.

1.2.4. Списочное декодирование

Декодирование линейных кодов по максимуму правдоподобия в общем случае является NP-полной задачей [42, 3, 81]. В большинстве случаев приходится строить приближенные алгоритмы решения этой задачи. Одним из часто используемых приемов является списочное декодирование, которое состоит в поиске L кодовых слов ближайших в некотором смысле к принятой последовательности [59]. Далее из полученного списка может быть выбрано

наиболее вероятное кодовое слово или кодовое слово, удовлетворяющее некоторым дополнительным проверочным соотношениям, которые не были учтены при построении этого списка. Сложность такого подхода определяется максимально возможным размером списка. В общем случае число кодовых слов кода длины n с минимальным расстоянием d над $\mathbb{F}_q$, находящихся в Хеммингговом шаре радиуса e, удовлетворяет (граница Джонсона [77])

$$J_q(n,d,e) \le \min\left\{ n(q-1), \frac{nd}{nd - 2e(n - \frac{qe}{2(q-1)})} \right\},$$

если

$$\frac{e}{n} < \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}\frac{d}{n}}\right).$$

Для некоторых кодов (например, кодов Рида-Маллера и сложенных кодов Рида-Соломона) $J_q(n,d,e)$ является полиномиальной функцией при значениях e, достигающих $d - \epsilon n$ для достаточно малых ϵ [75, 54, 43, 78].

Граница Джонсона может быть обобщена на случай мягкого декодирования [55]. Можно показать, что если зашумленные векторы y_0^{n-1} отнормировать так, чтобы выполнялось $\sum_{i=0}^{n-1} y_i^2 = n$, то число кодовых слов кода длины n с минимальным расстоянием d над алфавитом $\{1,-1\}$, удовлетворяющих

$$\sum_{i:y_i c_i<0} |y_i| \le T$$

при

$$T < \frac{1}{2}(W - \sqrt{(n-2d)n}),$$

не превосходит $\frac{2dn}{(W-2T)^2-(n-2d)n}$, где $W = \sum_{i=0}^{n-1} |y_i|$. Алгоритм мягкого списочного декодирования кодов Рида-Маллера с радиусом, удовлетворяющим этой границе, также приведен в [55]. Однако эксперименты показывают, что в случае канала с аддитивным белым гауссовским шумом для кодов с практически значимыми параметрами возникает необходимость декодирования (с нахождением единственного кодового слова) далеко за границу Джонсона.

1.2.5. Каналы передачи информации

В данной работе рассматриваются следующие модели каналов передачи информации:

1. Аддитивный гауссовский канал с двоичной фазовой модуляцией, для которого принятый сигнал может быть представлен как $y = (-1)^c + \eta, \eta \sim \mathcal{N}(0,\sigma^2)$, т.е.

$$W_{Y|X}(y|c) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-(-1)^c)^2}{2\sigma^2}}.$$

Декодирование двоичного кода $\mathcal{C}$ по максимуму правдоподобия для такого канала сводится к поиску кодового слова

$$\hat{c} = \arg\min_{c\in\mathcal{C}} \sum_{i=0}^{n-1} (y_i - (-1)^{c_i})^2 = \arg\max_{c\in\mathcal{C}} \sum_{i=0}^{n-1} y_i(-1)^{c_i} = \arg\min_{c\in\mathcal{C}} E(c_0^{n-1}, y_0^{n-1}),$$

где

$$E(c_0^{n-1}, y_0^{n-1}) = \sum_{i:y_i(-1)^{c_i}<0} |y_i|$$

— корреляционная невязка, известная также как эллипсоидальный вес вектора $c_0^{n-1} \in \mathbb{F}_2^n$ [183, 122].

2. q-ичный стирающий канал, для которого

$$W_{Y|X}\{y|x\} = \begin{cases} p, & y = \epsilon \\ 1-p, & y = x \\ 0, & y \notin \{x, \epsilon\}, \end{cases}$$

где p — вероятность стирания, ϵ — символ стирания и $x \in \mathbb{F}_q$. Декодирование по максимуму правдоподобия линейного кода в этом случае сводится к поиску решения системы линейных уравнений $\widetilde{H}\widetilde{c} = -\widehat{H}\widehat{c}$, где $\widehat{c}$ и $\widetilde{c}$ — подвекторы кодового слова, содержащие соответственно нестертые и стертые символы, а $\widetilde{H}, \widehat{H}$ — соответствующие подматрицы проверочной матрицы.

3. q-ичный симметричный канал, для которого

$$W_{Y|X}\{y|x\} = \begin{cases} 1-p, & y = x \\ \frac{p}{q-1}, & y \neq x, \end{cases}$$

для $y, x \in \mathbb{F}_q$, где p — вероятность ошибки. Декодирование кода $\mathcal{C}$ по максимуму правдоподобия для такого канала сводится к поиску $\widehat{c} = \arg\min_{c\in\mathcal{C}} d_H(c, y)$, где $d_H(c, y)$ — расстояние Хемминга между векторами $c, y \in \mathbb{F}_q^n$.

4. Канал с релеевскими замираниями с двоичной фазовой модуляцией, для которого $y = h(-1)^c + \eta, \eta \sim \mathcal{N}(0, \sigma^2)$ и h — случайная величина, распределенная по закону Релея[1]. Данная модель соответствует неселективному по частоте каналу с медленными замираниями и когерентным детектированием.

[1] Как правило, в этой модели предполагают, что h — комплексная гауссовская случайная величина. Однако этот случай может быть сведен к описанному выше путем использования когерентного приемника и достаточно большого перемежителя.

Все вышеперечисленные модели относятся к классу каналов без памяти.

Реальные каналы передачи информации описываются намного более сложными моделями (см., например, [74, 155, 150]). Они характеризуются наличием нескольких передающих и принимающих антенн, а также влиянием межсимвольной, межпользовательской, межканальной и межсотовой интерференции, причем параметры этой интерференции могут меняться со временем и могут не быть точно известны приемнику. Кроме того, следует учитывать случайные флуктуации отношения сигнал/шум, импульсные и узкополосные помехи, нелинейные искажения и др. Для организации передачи информации в каналах, подверженных этим явлениям, могут использоваться методы ортогонального частотного мультиплексирования, адаптивной передачи, прекодирования и др. Ввиду широкого распространения и высокой эффективности этих методов, во многих случаях при выборе корректирующих кодов и методов их декодирования ограничиваются рассмотрением простейших моделей каналов из вышеприведенного списка [65].

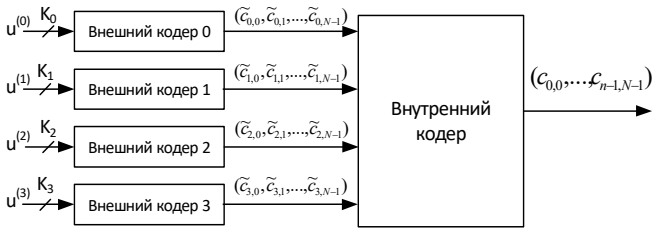
1.3. Обобщенные каскадные коды

1.3.1. Описание конструкции

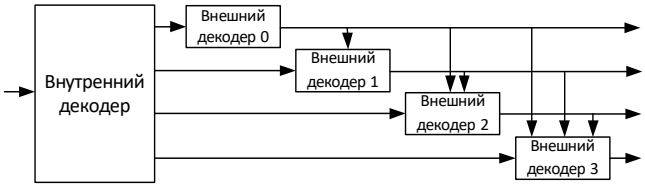
Пусть дано семейство внешних кодов $\mathbb{C}_i$ над $\mathbb{F}_{q^{m_i}}$ с параметрами $(N, K_i, D_i), 0 \le K_i \le N_i, 0 \le i < l$. Пусть $\mathcal{C}_0$ — внутренний код (в общем случае нелинейный) длины n, содержащий $\prod_{i=0}^{l-1} q^{m_i}$ кодовых слов. Будем считать, что внутренний код $\mathcal{C}_i^{(x)}$ представляет собой объединение q^{m_i} подкодов $\mathcal{C}_{i+1}^{(x,y)}, y \in \mathbb{F}_{q^{m_i}}, x \in \mathbb{F}_{q^{m_0}} \times \mathbb{F}_{q^{m_1}} \times \mathbb{F}_{q^{m_{i-1}}}, 0 \le i < l$, причем множества $\mathcal{C}_l^{(x_0,\ldots,x_{l-1})}$ содержат по одному элементу. Обобщенный каскадный код задается следующей процедурой кодирования [15]. Разобьем кодируемые данные на блоки размером $K_i m_i$ q-ичных символов. Зафиксировав некоторые базисы полей $\mathbb{F}_{q^{m_i}}$, представим их как блоки размером K_i символов из $\mathbb{F}_{q^{m_i}}$. Закодируем i-ый блок в коде $\mathbb{C}_i$ и запишем получившееся кодовое слово в i-ую строку $l \times N$ матрицы X. Заменим далее каждый столбец этой матрицы на единственный элемент $\mathcal{C}_l^{(X_{0,i},\ldots,X_{l-1,i})}$. В результате получим некоторое кодовое слово обобщенного каскадного кода (в общем случае нелинейного) длины Nn.

В тех случаях, когда в качестве внутреннего кода $\mathcal{C}_0$ используется сигнальное множество некоторого метода модуляции, данная конструкция носит название многоуровневого кода [88, 188].

В случае линейных внутренних кодов над некоторым полем Галуа описанная процедура кодирования может быть переформулирована следующим образом [7]. Пусть дано семейство вложенных внутренних линейных кодов $\mathcal{C}_i, 0 \le i < l$, над $\mathbb{F}_q$ с параметрами (n, k_i, d_i), причем



(*а*) Кодер



(*б*) Многошаговый декодер

Рис. 1.1. Кодирование и декодирование обобщенных каскадных кодов

$\mathcal{C}_0 \supset \mathcal{C}_1 \supset \cdots \supset \mathcal{C}_{l-1}$, $k_i - k_{i+1} = m_i, k_l = 0$. Пусть $\mathcal{G}$ — такая матрица, что ее последние k_i строк порождают $\mathcal{C}_i$. Обобщенный каскадный $(Nn, \sum_{i=0}^{l-1} K_i m_i, \geq \min_i D_i d_i)$ код над $\mathbb{F}_q$ задается следующей процедурой кодирования. Разобьем кодируемые данные на блоки размером $K_i m_i$ q-ичных символов. Зафиксировав некоторые базисы полей $\mathbb{F}_{q^{m_i}}$, представим их как блоки размером K_i символов из $\mathbb{F}_{q^{m_i}}$. Закодируем i-ый блок в коде $\mathbb{C}_i$ и запишем получившееся кодовое слово в i-ую строку $l \times N$ матрицы X. Пусть $k_0 \times N$ матрица $\widetilde{X}$ получена из X путем замены каждого элемента в i-ой строке на его разложение (коэффициенты которого представлены в виде вектора-столбца) в некотором фиксированном базисе $\mathbb{F}_{q^{m_i}}$. Вычислим $\widehat{X} = \mathcal{G}^T \widetilde{X}$. Выписывая по столбцам все элементы $\widehat{X}$, получим кодовое слово рассматриваемого обобщенного каскадного кода. Данная процедура проиллюстрирована на рис. 1.1, *а*.

Далее ограничимся случаем линейных кодов с $m_i = 1, 0 \leq i < l$.

1.3.2. Декодирование

Декодирование многоуровневых и обобщенных каскадных кодов может быть выполнено с помощью многошагового алгоритма, представленного на рис. 1.1, *б*. Алгоритм включает l шагов. Будем считать, что к i-ому шагу найдены правильные кодовые слова $(c_{s,0}, \ldots, c_{s,n-1})$ внешних кодов $\mathbb{C}_s, 0 \leq s < i$, соответствующие принятой последовательности. Тогда символы c_{sj} этих кодовых слов, $0 \leq j < N$, задают некоторый подкод $\mathcal{C}_i^{(c_{0,j},\ldots,c_{i-1,j})}$ кода $\mathcal{C}_0$ (в случае линейных кодов — смежные классы кода $\mathcal{C}_i$). На i-ом шаге данного алгоритма осуществляется расчет логарифмических отношений правдоподобия для символов кодового слова i-ого внешнего кода, т.е. информационных символов $k_0 - k_i, \ldots, k_0 - k_{i+1} - 1$ внутреннего кода $\mathcal{C}_0$.

Эта задача сводится к декодированию внутреннего кода с мягким выходом. При $q = 2$ она может быть сформулирована как

$$L_j = \log \frac{\sum_{z \in \mathcal{C}_{i+1}^{(c_{0,j},\ldots,c_{i-1,j},0)}} \prod_{s=0}^{n-1} W_{Y|X}(y_{jn+s}|z_s)}{\sum_{z \in \mathcal{C}_{i+1}^{(c_{0,j},\ldots,c_{i-1,j},1)}} \prod_{s=0}^{n-1} W_{Y|X}(y_{jn+s}|z_s)}.$$

Указанные значения могут быть вычислены с помощью расширенной решетки кода $\mathcal{C}_i$ [11]. После их нахождения производится декодирование внешнего кода $\mathbb{C}_i$. Многошаговый алгоритм не обеспечивает декодирование по максимуму правдоподобия. Причиной этого является то, что на i-ом шаге для исправления ошибок не используются проверочные символы кодов $\mathbb{C}_{i+1}, \ldots, \mathbb{C}_{l-1}$, и в случае ошибки декодирования внешнего кода она не может быть исправлена на последующих шагах.

Для преодоления этой проблемы могут использоваться методы итеративного [37, 192, 90] и списочного декодирования [79].

1.3.3. Выбор компонентных кодов

1.3.3.1. Правило пропускных способностей

Пусть X_i — случайная величина, соответствующая i-ому информационному символу внутреннего кода $\mathcal{C}_0$. Пусть Y — (векторная) случайная величина, соответствующая результату передачи кодового слова $\mathcal{C}_0$ по каналу без памяти. Заметим, что условная средняя взаимная информация этих случайных величин удовлетворяет

$$I(Y; X_i|X_0^{i-1}) = I(X_i; Y|X_0^{i-1}) = I(Y, X_0^{i-1}; X_i) - I(X_0^{i-1}; X_i).$$

В силу того, что кодовые слова различных внешних кодов выбираются независимо друг от друга, последний член в этом выражении равен 0. Таким образом, можно считать, что величины X_i передаются по синтетическому подканалу, на выходе которого наблюдаются значения $Y, X_0, \ldots, X_{i-1}$. Первая из этих величин соответствует выходу реального канала, а прочие соответствуют символам на выходе кодеров кодов $\mathbb{C}_i$. В реальности эти величины в приемнике недоступны, но они могут быть заменены выходом соответствующих декодеров в предположении, что декодирование на предшествующих шагах многошагового алгоритма было правильным. В силу того, что ошибка на любом из шагов этого алгоритма является фатальной, данное предположение не влияет на результаты анализа.

Для обеспечения сколь угодно надежной передачи информации в качестве кода $\mathbb{C}_i$ должен быть выбран код, достигающий пропускной способности соответствующего канала [188].

Скорость такого кода должна удовлетворять

$$R_i = \frac{K_i}{N_i} < I(Y, X_0^{i-1}; X_i) = I(X_i; Y|X_0^{i-1}).$$

Несложно убедиться в том, что

$$I(X_i; Y|X_0, \ldots, X_{i-1}) = \mathbf{M}_{X_0,\ldots,X_{i-1}}\left[C(\mathcal{C}_i^{(X_0,\ldots,X_{i-1})})\right] - \mathbf{M}_{X_0,\ldots,X_i}\left[C(\mathcal{C}_i^{(X_0,\ldots,X_i)})\right], \tag{1.4}$$

где $C(A)$ — пропускная способность канала при использовании сигнального множества A, причем оптимальное распределение X_i определяется оптимальным совместным распределением $X_0, \ldots, X_{l-1}$ для данного канала. В случае линейных кодов и двоичного симметричного канала расчет пропускных способностей при использовании соответствующих множеств рассмотрен в [47, 114].

Если все компонентные коды обеспечивают сколь угодно надежную передачу информации, то и соответствующий многоуровневый/обобщенный каскадный код также обеспечивает сколь угодно надежную передачу информации при скорости, меньшей пропускной способности $I(Y; X_0, \ldots, X_{l-1})$ [188]. Данное правило выбора параметров компонентных кодов носит название правила пропускных способностей. Следует отметить, что на практике приходится использовать коды конечной длины, которые должны иметь скорость, несколько меньшую по сравнению с пропускной способностью исходного канала. В связи с этим непосредственное применение правила пропускных способностей может приводить к неудовлетворительным кодам.

Вместе с тем, существуют и другие комбинации скоростей компонентных кодов, которые могут обеспечить достижение пропускной способности исходного канала. Однако при этом требуется использование декодера по максимуму правдоподобия, сложность которого, как правило, оказывается чрезмерно высокой [188].

1.3.3.2. Правило сбалансированных минимальных расстояний

Для максимизации размерности обобщенного каскадного кода при заданном его минимальном расстоянии d целесообразно выбирать минимальные расстояния компонентных кодов из условия [7, 188]

$$D_i d_i \approx d.$$

В [8, 79] показано, что выбирая в качестве внутренних кодов $\mathcal{C}_i$ коды, лежащие на границе Варшамова-Гилберта, а в качестве внешних — коды Рида-Соломона, данное правило позволяет при $\mathcal{N} \to \infty$ получить с полиномиальной сложностью коды $(\mathcal{N}, R\mathcal{N}, \mathcal{D})$, где $\mathcal{D}/\mathcal{N} \geq \delta$

и

$$R = \max_{0<r<1-H_q(\delta)} r - \frac{r}{s}\sum_{i=0}^{l-1}\frac{\delta}{H_q^{-1}(1-r+ri/l)}, \tag{1.5}$$

где

$$H_q(x) = x\log_q(q-1) - x\log_q(x) - (1-x)\log_q(1-x).$$

Однако вероятность ошибки декодирования определяется не только минимальным расстоянием, но и числом кодовых слов веса d, а также иными компонентами весового спектра кода. Практика показывает, что коды, построенные с использованием этого правила, имеют, как правило, неудовлетворительную вероятность ошибки многошагового декодирования. При этом оказывается, что скорости некоторых внешних кодов существенно превышают пропускные способности соответствующих каналов.

1.3.3.3. Правило экспоненты случайного кодирования

Как правило, от разработчика системы передачи информации требуется минимизировать вероятность ошибки для заданного канала при фиксированной скорости кодирования. В связи с этим можно предположить, что внешние коды имеют свойства, близкие к свойствам случайных кодов, а их декодирование осуществляется по максимуму правдоподобия. Скорости внешних кодов $R_i = K_i/N$ целесообразно выбирать так, чтобы их вероятность ошибки декодирования была примерно одинаковой [188], т.е. из условия

$$\max_{0\le\rho\le1}(E_0^i(\rho) - \rho R_i) = -\frac{1}{N}\log_2 p_w = const,$$

где

$$E_0^i(\rho) = -\mathbf{M}_{X_0,\ldots,X_{i-1}}\left[\log_2\left(\int_{\mathcal{Y}}\left(\sum_{x_i} W_X\{x_i\}\left(W_{Y|X_0^i}(y|X_0^{i-1},x_i)\right)^{\frac{1}{1+\rho}}\right)^{1+\rho}dy\right)\right]$$

— экспонента случайного кодирования для i-го синтетического подканала многоуровневого кода

$$W_{Y|X_0^i}(y|x_0^i) = \frac{1}{P\left\{\mathcal{C}_{i+1}^{(x_0^i)}\right\}}\sum_{c\in\mathcal{C}_{i+1}^{(x_0^i)}} P\{c\}\,W_{Y|X}(y|c),$$

W_X — распределение символов на входе этого подканала и $P\left\{\mathcal{C}_{i+1}^{(x_0^i)}\right\}$ — вероятность выбора некоторого подкода внутреннего кода. На практике, однако, сложность декодирования случайных кодов по максимуму правдоподобия может быть чрезмерно высокой, а реальный весовой спектр используемых кодов может быть весьма далек от среднего спектра случайных кодов.

1.3.3.4. Правило предельных вычислительных скоростей

Для учета ограничений на сложность декодирования внешних кодов можно воспользоваться понятием предельной вычислительной сложности, т.е. максимальной скорости сверточного кода, при которой средняя сложность последовательного декодирования остается конечной [92]. Можно показать, что эта скорость равна $E_0(1)$. Таким образом, скорость внешних кодов в обобщенном каскадном коде может быть выбрана равной $R_i = E_0^i(1)$ [188]. Однако сверточные коды асимптотически плохи и проигрывают по вероятности ошибки декодирования многим другим известным кодам, к алгоритмам декодирования которых понятие предельной вычислительной сложности, как правило, неприменимо.

1.3.3.5. Правило равных вероятностей ошибки

Практически наиболее полезным является правило равных вероятностей ошибки, которое предполагает, что скорости внешних кодов должны быть выбраны таким образом, чтобы обеспечить примерно одинаковую вероятность ошибки их декодирования в соответствующих синтетических каналах [188]. Однако в общем случае для практически применяемых алгоритмов декодирования отсутствуют точные формулы для расчета этих вероятностей ошибки, а известные верхние границы вероятности ошибки декодирования по максимуму правдоподобия требуют знания весовых спектров соответствующих кодов. Ввиду того, что нахождение весового спектра линейного кода является вычислительно крайне трудоемкой задачей, при использовании данного правила приходится полагаться на результаты статистического моделирования.

1.3.3.6. Коды для списочного декодирования

В [79] была предложена конструкция ОКК, допускающая списочное декодирование с полиномиальной сложностью с исправлением доли ошибок $\delta - \epsilon$ и скоростью, задаваемой (1.5). В основе этой конструкции лежат внутренние коды, допускающие эффективное списочное декодирование, и внешние сложенные коды Рида-Соломона, допускающие эффективное списочное восстановление (т.е. нахождение списка кодовых слов по списку возможных значений отдельных их символов).

1.4. Полярные коды

1.4.1. Поляризация канала

Пусть дана обратимая $l \times l$ матрица F_l над $\mathbb{F}_q$, называемая ядром. Наиболее известным примером ядра является матрица Арикана $F_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$.

Определение 1. $(n = l^m, k)$ *полярным кодом называется линейный блоковый код, порождаемый* k *строками матрицы поляризующего преобразования* $A_m = B_{l,m} F_l^{\otimes m}$*, где* $B_{l,m}$ *— матрица, задающая перестановку "обращения цифр", т.е. отображение* $i = \sum_{j=0}^{m-1} i_j l^j, 0 \le i_j < l$*, в* $\sum_{j=0}^{m-1} i_j l^{m-1-j}$.

Таким образом, кодовое слово полярного кода равно $c_0^{n-1} = u_0^{n-1} A_m$, где $u_i = 0, i \in \mathcal{F}$, и $\mathcal{F} \subset \{0, \dots, n-1\}$ — множество $n-k$ номеров *замороженных* символов. Конструкция *классических* полярных кодов подразумевает, что конкретные строки матрицы A_m, используемые в порождающей матрице полярного кода, выбираются из соображений минимизации вероятности ошибки декодирования нижеописанным методом последовательного исключения (ПИ) [30, 102, 126]. Пусть символы c_i передаются по симметричному каналу без памяти с q-ичным входом $\mathcal{W}_0^{(0)}(y|c) = W_{Y|X}(y|c)$. Определим синтетические символьные (при $q = 2$ — битовые) подканалы с функциями переходных вероятностей

$$\mathcal{W}_m^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i) = \frac{1}{q^{n-1}} \sum_{u_{i+1}^{n-1} \in \mathbb{F}_q^{n-i-1}} \prod_{j=0}^{n-1} \mathcal{W}_0^{(0)}(y_j|(u_0^{n-1} A_m)_j). \tag{1.6}$$

Здесь $a_s^t = (a_s, \dots, a_t)$, и $y_0, \dots, y_{n-1}$ — зашумленные символы, полученные в результате передачи символов $c_0, \dots, c_{n-1}$ по каналу $\mathcal{W}_0^{(0)}(y|u)$. Можно показать, что любое ядро вида $F_l' = U F_l P$, где U — обратимая верхнетреугольная матрица и P — перестановочная матрица, приводит к эквивалентным каналам $\mathcal{W}_m^{(i)}$. В связи с этим такие ядра можно считать эквивалентными и ввести понятие стандартной формы ядра, под которой понимается нижнетреугольная матрица с единичной диагональю $\tilde{F}_l = U F_l P$. Пусть $q = p^m$, где p — простое число. Можно показать, что если присоединение к полю $\mathbb{F}_p$ элементов матрицы $\tilde{F}_l$ приводит к полю $\mathbb{F}_q$, то матрица F_l является поляризующей, т.е. пропускные способности синтетических символьных подканалов сходятся с увеличением m сходятся к 0 или 1 q-ичных символов на одно использование канала, и доля подканалов с пропускными способностями близкими к 1 сходится к пропускной способности $\mathcal{W}_0^{(0)}(y|c)$ [126].

Удобно также ввести вероятности

$$\mathcal{W}_m^{(i)}\left\{u_0^i | y_0^{l^m-1}\right\} = \frac{\mathcal{W}_m^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i) P\{u_i\}}{\mathcal{W}(y_0^{n-1})},$$

где $\mathcal{W}(y_0^{n-1})$ — распределение символов на выходе канала, и $P\{u_i\} = 1/q, u_i \in \mathbb{F}_q$ — вероятности значений отдельных символов[2] на входе поляризующего преобразования. В общем случае они могут быть вычислены как

$$\mathcal{W}_\lambda^{(lj+i)}\left\{u_0^{lj+i}|y_0^{N-1}\right\} = \sum_{u_{lj+i+1}^{lj+l-1}\in\mathbb{F}_q^{l-i-1}} \prod_{s=0}^{l-1} \mathcal{W}_{\lambda-1}^{(j)}\left\{(u_{lt}^{lt+l-1}F_l)_s, 0 \le t \le j | y_{\frac{N}{l}s}^{\frac{N}{l}s+\frac{N}{l}-1}\right\} \tag{1.7}$$

для $u_i \in \mathbb{F}_q$, где $N = l^\lambda$. Можно заметить, что данная задача эквивалентна декодированию с мягким входом и мягким выходом в смежном классе кода, порождаемого строками $i, \ldots, l-1$ ядра F_l.

Алгоритм последовательного исключения на фазе i вычисляет $\mathcal{W}_m^{(i)}\{u_0^i|y_0^{n-1}\}$ и формирует решения

$$\widehat{u}_i = \begin{cases} \arg\max_{u_i\in\mathbb{F}_q} \mathcal{W}_m^{(i)}\{u_0^i|y_0^{n-1}\}, & i \notin \mathcal{F} \\ 0, & \text{иначе.} \end{cases} \tag{1.8}$$

Эти решения используются на последующих шагах вместо истинных значений u_i для оценивания $u_{i+1}, \ldots, u_{n-1}$. Можно показать [30], что при фиксированном l эти расчеты могут быть выполнены со сложностью $O(n\log n)$ операций вычисления (1.7). Сложность непосредственного вычисления значений этого выражения при условии того, что значения $\mathcal{W}_{m-1}^{(j)}(\cdot)$ известны, составляет $O(q^{l-i})$. Для упрощения вычислений может быть использован метод [11], основанный на использовании расширенной решетки соответствующего кода, порождаемого последними $i+1$ строками матрицы F. Однако данный подход не позволяет переиспользовать промежуточные результаты при последовательных попытках декодирования вложенных кодов уменьшающихся размерностей.

Для $q = 2$ параметры Бхаттачарьи $Z_{m,i}$ подканалов $\mathcal{W}_m^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i)$ удовлетворяют [102]

$$Z_{m-1,i}^{\Delta_j} \le Z_{m,il+j} \le 2^{l-j} Z_{m-1,i}^{\Delta_j}, 0 \le j < l, \tag{1.9}$$

где $Z_{0,0}$ — параметр Бхаттачарьи исходного канала, и Δ_j — частичные расстояния матрицы F_l. Под j-ым частичным расстоянием Δ_j матрицы F_l понимается минимальное расстояние Хемминга от ее j-ой строки до элементов кода, порождаемого строками с $j+1$ по $l-1$. Аналогичные оценки получены в [126] для случая $q > 2$.

Пусть $P_i \le (q-1)Z_{m,i}$ — вероятность неправильного оценивания символа u_i при использовании метода ПИ. Можно показать, что если матрица F_l является поляризующей, то для

[2] Очевидно, что это равенство не выполняется для случая замороженных символов. Однако символьные подканалы определяются для произвольных векторов u_0^i, и соответствующие вероятности вычисляются только для допустимых значений этих векторов.

всех $\epsilon > 0$

$$\lim_{m\to\infty} P\left\{P_J < 2^{-l^{(E(F_l)-\epsilon)m}}\right\} = C$$
$$\lim_{m\to\infty} P\left\{P_J < 2^{-l^{(E(F_l)+\epsilon)m}}\right\} = 0,$$

где J — случайная величина, равномерно распределенная на $\{0,\ldots,l^m-1\}$, C — пропускная способность канала $W_{Y|X}(y|x)$ и

$$E(F_l) = \frac{1}{l}\sum_{i=0}^{l-1}\log_l \Delta_i$$

— скорость поляризации ядра F_l [32, 102, 126].

Вероятность ошибки декодирования методом ПИ может быть оценена как

$$P \le \sum_{i\notin\mathcal{F}} P_i \tag{1.10}$$

Классический $(n = l^m, k)$ полярный код получается путем выбора $\mathcal{F}$ как множества $n-k$ номеров i, соответствующих наибольшим значениям P_i. Алгоритм расчета P_i, которые обладал бы полиномиальной сложностью, в общем случае пока неизвестен.

Алгоритм ПИ не обеспечивает декодирование по максимуму правдоподобия, т.к. на шаге i не учитывает ограничения замораживания, наложенные на символы $u_j, j > i$. Кроме того, даже при отсутствии таких ограничений он может принять решение в пользу значения $\hat{u}_i$, соответствующего большому числу маловероятных векторов $\hat{u}_0^i.v_{i+1}^{n-1}$, для которых $\mathcal{W}_m^{(i)}(\hat{u}_0^i|y_0^{n-1}) = \sum_{v_{i+1}^{n-1}}\mathcal{W}_m^{(n-1)}(\hat{u}_0^i.v_{i+1}^{n-1}|y_0^{n-1})$ больше вероятности вектора $\hat{u}_0^{n-1}$, который мог бы быть найден декодером МП, хотя каждая из вероятностей в этой сумме меньше вероятности вектора $\hat{u}_0^{n-1}$.

1.4.2. Ядра

1.4.2.1. Ядро Арикана

Наибольшую практическую значимость имеет случай $l = 2, q = 2$ и $F_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$. Выражение (1.7) при этом может быть переписано как

$$\mathcal{W}_\lambda^{(2i)}\left\{u_0^{2i}|y_0^{n-1}\right\} = \sum_{u_{2i+1}=0}^{1}\mathcal{W}_{\lambda-1}^{(i)}\left\{u_{0,e}^{2i+1}\oplus u_{0,o}^{2i+1}|y_0^{\frac{n}{2}-1}\right\}\mathcal{W}_{\lambda-1}^{(i)}\left\{u_{0,o}^{2i+1}|y_{\frac{n}{2}}^{n-1}\right\} \tag{1.11}$$

$$\mathcal{W}_\lambda^{(2i+1)}\left\{u_0^{2i+1}|y_0^{n-1}\right\} = \mathcal{W}_{\lambda-1}^{(i)}\left\{u_{0,e}^{2i+1}\oplus u_{0,o}^{2i+1}|y_0^{\frac{n}{2}-1}\right\}\mathcal{W}_{\lambda-1}^{(i)}\left\{u_{0,o}^{2i+1}|y_{\frac{n}{2}}^{n-1}\right\}, \tag{1.12}$$

где $u_{0,e}^j$ и $u_{0,o}^j$ — подвекторы вектора u_i^j, образованные его элементами с четными и нечетными номерами, соответственно. Эти формулы удобно переписать в терминах логарифмических

отношений правдоподобия

$$L_{\lambda}^{(2i)}(u_0^{2i-1}|y_0^{N-1}) = 2\tanh^{-1}\left(\tanh\left(\frac{L_{\lambda-1}^{(i)}(u_{0,e}^{2i-1}\oplus u_{0,o}^{2i-1}|y_0^{\frac{N}{2}-1})}{2}\right)\cdot \right.$$
$$\left.\cdot\tanh\left(\frac{L_{\lambda-1}^{(i)}(u_{0,o}^{2i-1}|y_{\frac{N}{2}}^{N-1})}{2}\right)\right), \tag{1.13}$$

$$L_{\lambda}^{(2i+1)}(u_0^{2i}|y_0^{N-1}) = (-1)^{u_{2i}}L_{\lambda-1}^{(i)}(u_{0,e}^{2i-1}\oplus u_{0,o}^{2i-1}|y_0^{\frac{N}{2}-1}) + L_{\lambda-1}^{(i)}(u_{0,o}^{2i-1}|y_{\frac{N}{2}}^{N-1}), \tag{1.14}$$

где $L_{\lambda}^{(i)}(u_0^{i-1}|y_0^{N-1}) = \log\frac{\mathcal{W}_{\lambda}^{(i)}\{u_0^{i-1}.0|y_0^{N-1}\}}{\mathcal{W}_{\lambda}^{(i)}\{u_0^{i-1}.1|y_0^{N-1}\}}$, $0\le i<n, 0\le\lambda<m$. Тогда правило принятия решений в алгоритме последовательного исключения для $i\notin\mathcal{F}$ может быть записано как

$$\hat{u}_i = \begin{cases} 0, & L_m^{(i)}(u_0^{i-1}|y_0^{n-1}) > 0 \\ 1, & \text{иначе.} \end{cases}$$

Параметры Бхаттачарьи [30, 86] битовых подканалов, порождаемых поляризующим преобразованием Арикана, удовлетворяют

$$Z_{\lambda-1,i}\sqrt{2-Z_{\lambda-1,i}^2} \le Z_{\lambda,2i} \le 2Z_{\lambda-1,i} - Z_{\lambda-1,i}^2 \tag{1.15}$$

$$Z_{\lambda,2i+1} = Z_{\lambda-1,i}^2. \tag{1.16}$$

В случае двоичного стирающего канала справедливо также

$$Z_{\lambda,2i} = 2Z_{\lambda-1,i} - Z_{\lambda-1,i}^2. \tag{1.17}$$

Вероятности ошибки P_i в синтетических подканалах могут быть вычислены с помощью метода эволюции плотностей [124], реализация которого со сложностью $O(n\mu^2\log\mu)$ описана в [158], где μ — число уровней квантования, увеличение которого приводит к увеличению точности вычисления P_i.

При использовании алгоритма последовательного исключения минимальная длина классического полярного кода со скоростью R, обеспечивающего заданную вероятность ошибки в канале с пропускной способностью C, удовлетворяет

$$\frac{\alpha}{(C-R)^{3.579}} \le n_0 \le \frac{\beta}{(C-R)^6},$$

для некоторых α,β, зависящих от канала и вероятности ошибки, что значительно хуже, чем для случайных кодов (см. раздел 1.2.3) [85]. Причиной этого является то, что поляризация битовых подканалов происходит достаточно медленно, т.е. при построении классического полярного кода приходится оставлять достаточно большое число незамороженных символов,

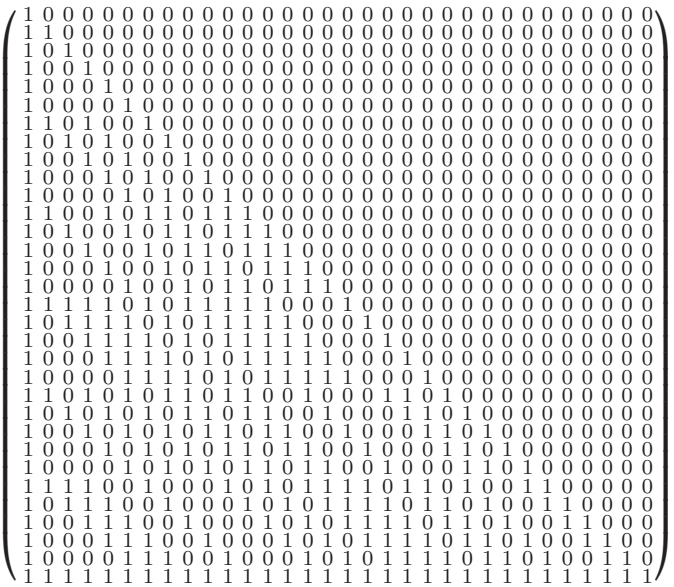


Рис. 1.2. Расширенное ядро БЧХ F_{32}

которые передаются по битовым подканалам с высокой вероятностью ошибки. Следствием этого является то, что вероятность ошибки декодирования полярных кодов умеренных длин (до нескольких тысяч) даже при использовании описанного ниже списочного метода декодирования оказывается существенно выше, чем у известных кодов с малой плотностью проверок на четность и турбо-кодов с аналогичными параметрами.

1.4.2.2. Ядро БЧХ

Как известно, коды БЧХ обладают свойством вложенности. В связи с этим, при $l = 2^\mu$ расширенное ядро БЧХ может быть получено как матрица F_l, где $((F_l)_{i+1,1}, \ldots, (F_l)_{i+1,l-1})$ — вектор коэффициентов многочлена $x^j g_{i'}(x)$, $g_{i'}(x)$ — порождающий многочлен $(l-1, l-1-i')$ кода БЧХ, и j — наименьшее натуральное число, для которого $i = j+i'$. Кроме того, $(F_l)_{0,0} = 1$ и $(F_l)_{i+1,0} = \sum_{j=1}^{l-1} (F_l)_{i+1,j}$. На рис. 1.2 приведен пример ядра РБЧХ.

1.4.2.3. Ядро Рида-Соломона

При $l \le q$ наибольшая скорость поляризации достигается при использовании ядра Рида-Соломона[3] [126]

$$F_l = \begin{pmatrix} \alpha_{l-1}^{l-1} & \alpha_{l-2}^{l-1} & \dots & \alpha_2^{l-1} & \alpha_1^{l-1} & \alpha_0^{l-1} \\ \alpha_{l-1}^{l-2} & \alpha_{l-2}^{l-2} & \dots & \alpha_2^{l-2} & \alpha_1^{q-2} & \alpha_0^{l-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \alpha_{l-1}^{1} & \alpha_{l-2}^{1} & \dots & \alpha_2^{1} & \alpha_1^{1} & \alpha_0^{1} \\ \alpha_{l-1}^{0} & \alpha_{l-2}^{0} & \dots & \alpha_2^{0} & \alpha_1^{0} & \alpha_0^{0} \end{pmatrix}, \tag{1.18}$$

где α_i — различные элементы $\mathbb{F}_q$.

Можно показать, что в случае q-ичного стирающего канала с вероятностью стирания ϵ для достаточно больших q и всех $\gamma, \delta > 0$ доля подканалов, задаваемых поляризующим преобразованием Рида-Соломона с ядром F_q, с вероятностью стирания не более $O(n^{-\gamma})$ ограничена снизу величиной $1 - \epsilon - O(n^{-1/2+\delta})$, т.е. экспонента масштабирования полярных кодов с ядром Рида-Соломона близка к 2 [137]. Следует отметить, что это достигается за счет увеличения размера алфавита, в то время как случайные коды обеспечивают экспоненту масштабирования 2 уже для двоичного алфавита.

Ядро Арикана является частным случаем ядра Рида-Соломона при $l = q = 2$.

1.4.3. Полярные коды как обобщенные каскадные

Для любого $s, 0 < s \le m$, полярные коды могут рассматриваться как обобщенные каскадные коды с внутренними кодами, порождаемыми строками матрицы $F_l^{\otimes s}$, и внешними q-ичными кодами, порождаемыми некоторыми подматрицами $F_l^{\otimes(m-s)}$. Более конкретно, если $\mathcal{F}$ — множество номеров замороженных символов для кода длины $n = l^m$, то i-ый внешний код является полярным кодом с множеством номеров замороженных символов

$$\mathcal{F}^{(i)} = \left\{ j \bmod l^{m-s} \middle| \left\lfloor j / l^{m-s} \right\rfloor = i, j \in \mathcal{F} \right\}, \tag{1.19}$$

а i-ый внутренний код является полярным с множеством номеров замороженных символов $\{0, \dots, i-1\}$. Это свойство может быть использовано для упрощения декодирования полярных кодов и их аналогов.

[3] Эту конструкцию можно обобщить на случай $l = q + 1$, используя конструкцию дважды расширенных кодов Рида-Соломона [18].

1.4.4. Списочный метод последовательного исключения

1.4.4.1. Алгоритм Тала-Варди

Алгоритм последовательного исключения не обеспечивает декодирование полярных кодов по максимуму правдоподобия. Причиной этого является то, что правило (1.8) не учитывает ограничения замораживания, наложенные на символы с номерами $j > i, j \in \mathcal{F}$, и в случае формирования неверной оценки $\hat{u}_i$ эта ошибка не может быть далее исправлена.

В работе [159] был предложен списочный метод последовательного исключения. Его идея состоит в том, что декодер хранит не более чем L частичных векторов (путей) v_0^{i-1}. На каждой фазе формируются не более чем $2L$ всевозможных продолжений v_0^i этих путей с учетом ограничений замораживания. Из них выбираются L путей с наибольшими значениями $\mathcal{W}_m^{(i)}(v_0^i|y_0^{n-1})$ (вес пути), а остальные пути исключаются из рассмотрения. В качестве решения задачи декодирования выбирается путь v_0^{n-1} с наибольшим значением $\mathcal{W}_m^{(n-1)}(v_0^{n-1}|y_0^{n-1})$. При надлежащем выборе структур данных эти вычисления могут быть выполнены со сложностью $O(Ln\log n)$. Для классических полярных кодов Арикана применение такого подхода позволяет реализовать декодирование почти по максимуму правдоподобия уже для небольших L ($L \approx 16$) [159]. Более того, было замечено, что правильное кодовое слово практически всегда присутствует в списке, формируемом этим алгоритмом. В связи с этим в [159] было предложено добавлять к данным, кодируемым полярным кодом, контрольную сумму CRC, значение которой можно использовать для выбора из списка решения задачи декодирования. Полученные таким образом коды имеют существенно лучшую корректирующую способность по сравнению с классическими полярными кодами.

Существенным недостатком алгоритма Тала-Варди является то, что он всегда формирует L кодовых слов полярного кода, из которых правильным решением задачи декодирования может быть не более чем одно. Для декодирования по максимуму правдоподобия полярных кодов с CRC и других улучшенных конструкций может требоваться весьма большой размер списка L. Таким образом, значительный объем вычислений выполняется впустую, но существует потенциальная возможность существенного снижения сложности.

1.4.4.2. Аппроксимация весовой функции

В работе [35] для списочного декодирования полярных кодов с ядром Арикана было предложено аппроксимировать значения $\log \mathcal{W}_m^{(i)}(v_0^i|y_0^{n-1})$ величиной

$$R(v_0^i, y_0^{n-1}) = R(v_0^{i-1}, y_0^{n-1}) + \tau(S_m^{(i)}(v_0^{i-1}, y_0^{n-1}), v_i), \tag{1.20}$$

где

$$\tau(S, v) = \begin{cases} 0, & \text{если } (-1)^v = \operatorname{sgn} S \\ -|S|, & \text{иначе} \end{cases}$$

— штрафная функция и $S_m^{(i)}(v_0^{i-1}, y_0^{n-1})$ — приближенные логарифмические отношения правдоподобия (ЛОПП), задаваемые

$$S_\lambda^{(2i)}(v_0^{2i-1}, y_0^{N-1}) = \operatorname{sgn}(a)\operatorname{sgn}(b)\min(|a|, |b|), \tag{1.21}$$

$$S_\lambda^{(2i+1)}(v_0^{2i}, y_0^{N-1}) = (-1)^{\hat{u}_{2i}} a + b, \tag{1.22}$$

причем $a = S_{\lambda-1}^{(i)}(v_{0,e}^{2i-1} \oplus v_{0,o}^{2i-1}, y_0^{\frac{N}{2}-1})$, $b = S_{\lambda-1}^{(i)}(v_{0,o}^{2i-1}, y_{\frac{N}{2}}^{N-1})$, и $S_0^{(0)}(y) = \log \frac{\mathbf{W}(y|0)}{\mathbf{W}(y|1)}$.

1.4.4.3. Вероятность ошибки

Вероятность ошибки декодирования с помощью метода Тала-Варди может быть оценена как

$$P(L) \le P_{ML} + P(E(L)|\mathbb{C}), \tag{1.23}$$

где P_{ML} — вероятность ошибки декодирования соответствующего кода по максимуму правдоподобия, $\mathbb{C}$ — событие, соответствующее принятию правильного решения $u_0^{n-1}A_m$ декодером макимума правдоподобия, и $E(L)$ — событие, соответствующее тому, что вес правильного пути u_0^i на некоторой фазе i оказывается меньше весов L неправильных путей $v_0^i \neq u_0^i$, что приводит к исключению декодером правильного пути из рассмотрения. Существуют достаточно точные границы для P_{ML}, использующие весовой спектр кода [140]. В слабо зашумленных каналах без памяти P_{ML} в основном зависит от минимального расстояния d и ошибочного коэффициента кода, т.е. числа w_d кодовых слов веса d. Насколько известно автору, до настоящего времени отсутствуют способы оценивания $P(E(L)|\mathbb{C})$. Вместе с тем, эксперименты свидетельствуют о том, что эта величина возрастает с увеличением P_{SC}, вероятности ошибки декодирования кода методом последовательного исключения.

Необходимо отметить, что существует возможность того, что списочный метод Тала-Варди найдет правильное кодовое слово $u_0^{n-1}A_m$ даже в том случае, когда декодер максимума правдоподобия совершает ошибку. Такое возможно, если путь, соответствующий ошибочному решению декодера максимума правдоподобия, исключается из рассмотрения на некоторой промежуточной фазе i, но правильный путь u_0^i остается в списке. Однако вероятность такого события пренебрежимо мала, и неравенство в (1.23) является достаточно точным.

1.5. Коды Рида-Соломона

1.5.1. Определения

Определение 2. $(n, k, d = n - k + 1)$ *кодом Рида-Соломона называется циклический код длины* $n < q$ *над* $\mathbb{F}_q$ *с порождающим многочленом* $g(x) = \prod_{i=0}^{n-k-1}(x - \alpha^{b+i})$, *где* α *— некоторый элемент* $\mathbb{F}_q$ *порядка* n.

При $n = q - 1$ получаемые коды называются примитивными, а при $b = 1$ — кодами Рида-Соломона в узком смысле. Код Рида-Соломона в смысле определения 2 является циклическим кодом Боуза-Чоудхури-Хоквингема (БЧХ).

Как и для любого циклического кода, систематическое кодирование в коде Рида-Соломона может быть выполнено как

$$c(x) = a(x)x^{n-k} + r(x), \tag{1.24}$$

где $r(x) \equiv -a(x)x^{n-k} \bmod g(x)$, $a(x) = \sum_{i=0}^{k-1} a_i x^i$, a_i — информационные символы, $c(x) = \sum_{i=0}^{n-1} c_i x^i$ — многочлен, коэффициентами которого являются символы кодового слова. В случае $q = 2^m$ сложность этого алгоритма может быть снижена, если перед делением на $g(x)$ выполнить деление на многочлен $\psi(x) \in \mathbb{F}_2[x]$ степени $o(n)$ и веса (т.е. с числом ненулевых коэффициентов) $O(d)$, делящийся на $g(x)$ [149].

Во многих случаях оказывается полезным приведенное ниже альтернативное определение кода Рида-Соломона.

Определение 3. $(n, k, n - k + 1)$ *кодом Рида-Соломона над* $\mathbb{F}_q$ *называется множество векторов* $(f(x_0), \ldots, f(x_{n-1}))$, *где* $f(x) = \sum_{i=0}^{k-1} f_i x^i, f_i \in \mathbb{F}_q$, x_i *— различные элементы* $\mathbb{F}_q$.

Приведенные определения кода Рида-Соломона эквивалентны при $n = q-1, b = 1, x_i = \alpha^i$.

Определение 4. $(n, k, n-k+1)$ *обобщенным кодом Рида-Соломона называется множество векторов* $(c_0\omega_0, \ldots, c_{n-1}\omega_{n-1})$, *где* ω_i *— некоторые фиксированные ненулевые элементы* $\mathbb{F}_q$ *и* $(c_0, \ldots, c_{n-1})$ *— кодовые слова некоторого* $(n, k, n - k + 1)$ *кода Рида-Соломона.*

1.5.2. Декодирование кодов Рида-Соломона

Известно, что декодирование кодов Рида-Соломона по максимуму правдоподобия является NP-полной задачей [81]. Оно остается таковым даже при ограничении радисуа декодирования до $n - k - O(\log n / \log\log n)$ [71]. В связи с этим приходится использовать субоптимальные методы декодирования с полиномиальной сложностью.

Классическая процедура декодирования кодов БЧХ (в т.ч. и Рида-Соломона) над $\mathbb{F}_q$ основана на использовании вектора синдрома

$$S_i = \sum_{j=0}^{n-1} y_j \alpha^{(b+i)j}, 0 \le i < d-1, \tag{1.25}$$

где d — конструктивное расстояние кода, $(y_0, \ldots, y_{n-1}) \in \mathbb{F}_q^n$ — декодируемый вектор, причем $y_i = c_i + e_i$, где c_i — символы кодового слова и e_i — значения ошибок. Введем многочлен значений ошибок и стираний

$$\Gamma(x) = \Lambda(x) S(x) \bmod x^{d-1}, \tag{1.26}$$

где $\Lambda(x) = \prod_{j=1}^{t}(1 - X_j x)$ — многочлен локаторов ошибок и стираний,

$$S(x) = \sum_{i=0}^{d-2} S_i x^i$$

— синдромный многочлен, $X_j = \alpha^{i_j}$ — локатор j-ой ошибки, i_j — номер j-го ошибочного символа, $E_j = e_{i_j}$, t — число произошедших ошибок. Заметим, что

$$S(x) = \sum_{j=1}^{t} E_j X_j^b \sum_{i=0}^{d-2} (X_j x)^i \equiv \sum_{j=1}^{t} \frac{E_j X_j^b}{1 - X_j x} \bmod x^{d-1}.$$

Отсюда видно, что значение j-ого ошибочного символа может быть найдено как

$$E_j = \frac{X_j^{-b} \Gamma(X_j^{-1})}{\prod_{l \ne j}(1 - X_l X_j^{-1})}. \tag{1.27}$$

Вводя формальную производную многочлена $\Lambda(x)$, получим

$$\Lambda'(x) = \sum_{i=1}^{t} i \Lambda_i x^{i-1} = \Lambda_t \sum_{s=1}^{t} \prod_{p \ne s} (x - X_p^{-1}).$$

Поэтому справедливо соотношение

$$\Lambda'(X_i^{-1}) = \underbrace{\left(\prod_{p=1}^{t} X_p\right)}_{\Lambda_t} \prod_{p \ne i} (X_i^{-1} - X_p^{-1}) = X_i \prod_{p \ne i} (X_p X_i^{-1} - 1),$$

откуда для полей характеристики 2 следует

$$E_j = \frac{X_j^{(1-b)}}{\sum_{s=0}^{\lfloor (t-1)/2 \rfloor} \Lambda_{2s+1} X_j^{-2s}} \Gamma(X_j^{-1}).$$

Из приведенных преобразований видно, что $\deg \Gamma(x) \le t - 1$. Однако ни значение t, ни локаторы ошибок неизвестны. Можно, однако, показать, что уравнение (1.26) имеет единственное решение (пару $\Lambda(x), \Gamma(x)$) при $t \le \lfloor (d-1)/2 \rfloor$. Оно может быть найдено или с помощью расширенного алгоритма Евклида, или с помощью алгоритма Берлекэмпа-Месси [5]. Действительно, из ограничений на степень $\Gamma(x)$ и определения $\Lambda(x)$ вытекает, что

$$\sum_{j=1}^{t} \Lambda_j S_{i+t-j} = -S_{i+t}, 0 \le i < d - t - 1. \tag{1.28}$$

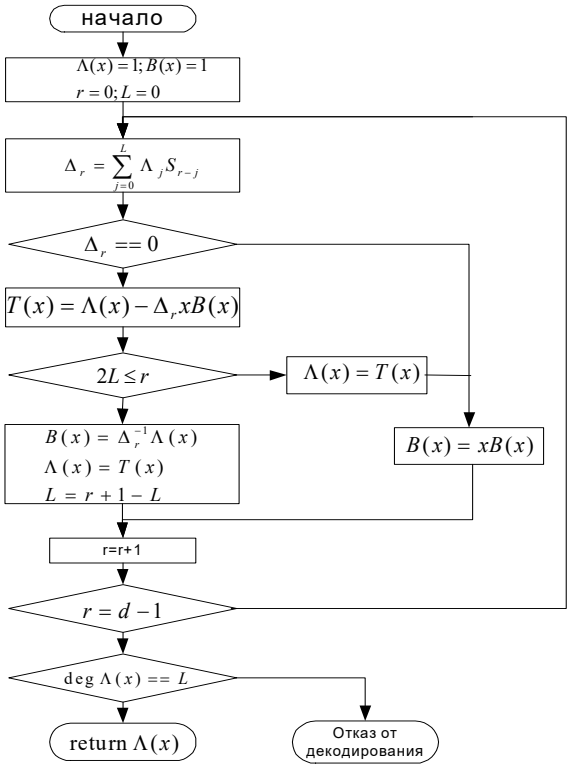


Рис. 1.3. Алгоритм Берекэмпа-Месси

Тогда коэффициенты $\Lambda(x)$ задают некоторый регистр сдвига с линейной обратной связью, порождающий последовательность S_i, а сам $\Lambda(x)$ называют многочленом связей этого регистра. Эти коэффициенты могут быть найдены с помощью алгоритма Берлекэмпа-Месси, приведенного на рис. 1.3.После нахождения $\Lambda(x)$ могут быть найдены его корни, которые задают местоположение ошибок.

Процедура вычисления синдрома, основанная на непосредственном применении (1.25), имеет сложность $O(dn)$, процедура поиска корней $\Lambda(x)$ с помощью процедуры Ченя (полного перебора возможных корней) имеет сложность $O(tn)$, в то время как сложность алгоритма Берлекэмпа-Месси имеет сложность $O(d^2)$. Таким образом, вычисление синдрома и поиск корней многочлена локаторов ошибок являются наиболее трудоемкими этапами классической процедуры декодирования кодов БЧХ (и, в частности, Рида-Соломона).

Описанная процедура позволяет найти единственное кодовое слово кода БЧХ (если оно существует), находящееся на расстоянии не более чем $\lfloor (d-1)/2) \rfloor$ от декодируемого вектора. Однако на практике зачастую требуется исправление намного большего числа ошибок. В этом случае декодирование может быть неоднозначным, т.е. декодер может возвратить список. Соответствующие алгоритмы будут рассмотрены ниже.

Отметим, что ключевое уравнение может быть использовано и для исправления стираний. В этом случае можно считать, что i_j — номер j-го стертого символа, $y_{i_j} = 0$ и найти

$\Gamma(x)$ непосредственно из (1.26).

1.5.3. Мономиальные упорядочения

При описании многих алгоритмов декодирования кодов Рида-Соломона оказывается полезным использование мономиальных упорядочений [49].

(a,b)-взвешенная степень одночлена x^iy^j по определению равна $ai+bj$. (a,b)-взвешенная степень многочлена $Q(x,y)$, обозначаемая $\mathrm{wdeg}_{(a,b)}\,Q(x,y)$, равна максимальной из (a,b)-взвешенных степеней его ненулевых членов. Взвешенная степень может быть использована для упорядочения одночленов. (a,b)-взвешенное лексикографическое упорядочение определяется как $x^iy^j \prec x^py^q \Leftrightarrow (ai+bj < ap+bq) \vee (ai+bj = ap+bq) \wedge (x^iy^j \prec_{lex} x^py^q)$. Лексикографическое упорядочение определяется как $x^iy^j \prec_{lex} x^py^q \Leftrightarrow (j<q) \vee (j=q) \wedge (i<p)$. Старшим мономом $\mathrm{LM}\,Q(x,y)$ многочлена $Q(x,y) = \sum q_{ij}x^iy^j$ называется $\max\limits_{q_{ij}\neq 0} x^iy^j$, где при выборе максимума используется некоторое мономиальное упорядочение, которое будет ясно далее из контекста. Старшим коэффициентом многочлена называется коэффициент при старшем мономе многочлена, а старшим членом $\mathrm{LT}\,Q(x,y)$ многочлена — произведение его старшего коэффициента и старшего монома.

На множестве многочленов можно ввести отношение строгого предпорядка, полагая, что

$$Q(x,y) \prec P(x,y) \Leftrightarrow \mathrm{LM}\,Q(x,y) \prec \mathrm{LM}\,P(x,y).$$

В дальнейшем запись $\min Z$ (или минимальный элемент Z), где Z — некоторое множество многочленов, будет обозначать нахождение такого многочлена $S(x,y) \in Z$, что $\forall T(x,y) \in Z : T(x,y) = S(x,y) \vee S(x,y) \prec T(x,y)$. Во всех случаях множество Z будет строиться так, чтобы обеспечить существование и единственность такого многочлена.

1.5.4. Метод Гурусвами-Судана списочного декодирования кодов Рида-Соломона

Рассмотрим задачу списочного декодирования $(n,k,n-k+1)$ кода Рида-Соломона над некоторым полем $\mathbb{F}$. Списочное декодирование может быть использовано для повышения корректирующей способности по сравнению с классическим методом декодирования, описанным в разделе 1.5.2. Кроме того, список кодовых слов, формируемый декодером, может представлять самостоятельную ценность в задачах стеганографии [67] и теории сложности [156].

Для произвольного вектора $(y_0,\ldots,y_{n-1})$ метод Гурусвами-Судана позволяет найти

все такие многочлены, $f(x)$ что $\deg f(x) < k$ и $f(x_i) = y_i$ для не менее чем τ различных x_i [80]. Это достигается путем построения такого многочлена $Q(x,y)$, что $Q(x_i,y_i) = 0^r, \mathrm{wdeg}_{(1,k-1)} Q(x,y) \le l$, $\mathrm{wdeg}_{(0,1)} Q(x,y) < \rho, 0 \le i < n$, и поиска его функциональных корней, т.е. таких многочленов $f(x)$, что $Q(x,f(x)) = 0$ и $\deg f(x) < k$. Здесь и далее запись $Q(x_i,y_i) = 0^r$ означает, что многочлен $Q(x,y)$ имеет корень (x_i,y_i) кратности r, т.е. все его частные производные Хассе

$$Q^{[a,b]}(x_i,y_i) = \sum_{j_1 \ge a} \sum_{j_2 \ge b} C_{j_1}^a C_{j_2}^b q_{j_1,j_2} x_i^{j_1-a} y_i^{j_2-b}$$

порядка $a+b < r$ равны 0. Можно показать, что все решения задачи списочного декодирования могут быть найдены из полученных $f(x)$, если параметры алгоритма удовлетворяют следующим условиям [129]:

$$\frac{\rho(\rho-1)}{2} \le \frac{nr(r+1)}{2(k-1)} < \frac{\rho(\rho+1)}{2}, \tag{1.29}$$

$$l = \left\lfloor \frac{nr(r+1)}{2\rho} + \frac{(\rho-1)(k-1)}{2} \right\rfloor, \tag{1.30}$$

$$\tau = \left\lfloor \frac{l}{r} \right\rfloor + 1 > \sqrt{n(k-1)}. \tag{1.31}$$

Следует отметить, что многочлен $Q(x,y)$, удовлетворяющий вышеприведенным условиям, может иметь функциональные корни $f(x)$, не соответствующие допустимым решениям задачи списочного декодирования. Кроме того, приведенный ниже алгоритм поиска функциональных корней может возвратить паразитные многочлены $f(x) : Q(x,f(x)) \ne 0$. В связи с этим для всех найденных $f(x)$ необходимо построить соответствующие кодовые слова и выбрать те из них, которые совпадают с принятым вектором не менее чем в τ позициях.

1.5.4.1. Интерполяция

Построение многочлена $Q(x,y)$ является наиболее вычислительно сложным этапом метода Гурусвами-Судана. Заметим, что множество многочленов $I_r = \{Q(x,y) \in \mathbb{F}[x,y] | Q(x_i,y_i) = 0^r, 1 \le i \le n\}$ образует идеал. Ненулевой многочлен этого идеала с минимальным старшим членом относительно $(1,k-1)$-взвешенного лексикографического упорядочения удовлетворяет ограничениям метода Гурусвами-Судана. Такой многочлен входит в базис Грёбнера идеала I_r относительно данного упорядочения [151]. Однако оказывается проще построить базис Грёбнера модуля $M_{r,\rho} = \{Q(x,y) \in I_r | \mathrm{wdeg}_{(0,1)} Q(x,y) < \rho\}$.

Итеративный интерполяционный алгоритм (ИИА), приведенный на рис. 1.4, формирует ρ таких ненулевых многочленов $Q_j(x,y), 0 \le j \le \rho-1$, что $Q(x_i,y_i) = 0^r, \mathrm{LT}\, Q_j(x,y) = x^{t_j} y^j$, причем t_j являются наименьшими возможными целыми числами [129, 97, 133].

IterativeInterpolation$(n, \{(x_i, y_i), 1 \le i \le n\}, r, \rho)$

for $i \leftarrow 0$ **to** $\rho - 1$
do $Q_i(x, y) \leftarrow y^i$;
for $i \leftarrow 1$ **to** n
do for $\beta \leftarrow 0$ **to** $r - 1$
  do for $\alpha \leftarrow 0$ **to** $r - \beta - 1$
    do $\Delta_j \leftarrow Q_j^{[\alpha,\beta]}(x_i, y_i), 0 \le j \le \rho - 1$
      $m \leftarrow \arg\min_{j:\Delta_j \neq 0} Q_j(x, y)$
      for $j \neq m$
      do $Q_j(x, y) \leftarrow Q_j(x, y) - \frac{\Delta_j}{\Delta_m} Q_{j_0}(x, y)$
      $Q_m(x, y) \leftarrow Q_m(x, y)(x - x_i)$;
return $(Q_0(x, y), \ldots, Q_{\rho-1}(x, y))$

Рис. 1.4. Итеративный интерполяционный алгоритм (ИИА)

Лемма 1. *Пусть $Q_j(x, y), 0 \le j \le \rho - 1$ — многочлены, построенные ИИА для заданного набора точек $(x_i, y_i), 1 \le i \le n$, параметров r и ρ. Тогда любой многочлен $Q(x, y)$, удовлетворяющий условиям $Q(x_i, y_i) = 0^r, 1 \le i \le n$, и $\operatorname{wdeg}_{(0,1)} Q(x, y) < \rho$, может быть представлен как*

$$Q(x, y) = \sum_{j=0}^{\rho-1} p_j(x) Q_j(x, y),$$

причем $\operatorname{LT} Q(x, y) = \operatorname{LT} p_j(x) \operatorname{LT} Q_j(x, y)$ для некоторого j. Другими словами, эти многочлены образуют базис Грёбнера модуля $M_{r,\rho}$ [112, 21].

Доказательство. Рассмотрим произвольный многочлен $Q(x, y) \in M_{r,\rho}$. Пусть $\operatorname{LT} Q(x, y) = \alpha x^a y^j$. Заменим $Q(x, y)$ на $Q(x, y) - \alpha x^{a-t_j} Q_j(x, y)$. Это можно сделать в силу того, что величины t_j являются минимальными возможными. Будем повторять такие действия до тех пор, пока это возможно. Полученный таким образом многочлен $Q'(x, y)$ является нулевым, т.к. в противном случае он принадлежал бы $M_{r,\rho}$ и обладал бы старшим членом $\operatorname{LT} Q'(x, y) = \beta x^b y^i : b < t_i$. Но известно, что t_i являются минимальными возможными. □

В контексте метода Гурусвами-Судана в ИИА должно использоваться $(1, k - 1)$-взвешенное лексикографическое упорядочение. Решение задачи интерполяции задается минимальным из построенных многочленов $(Q_0(x, y), \ldots, Q_{\rho-1}(x, y))$. Можно заметить, что сложность ИИА составляет $O(n^2 r^4 \rho)$ операций. Такую же сложность имеет алгоритм

RECONSTRUCT$(Q(x,y),k,i,\Phi)$

1 Найти наибольшее r, т.ч. $x^{-r}Q(x,y)$ является многочленом
2 $M(x,y) \leftarrow x^{-r}Q(x,y)$
3 Найти корни $\gamma_i \in \mathbb{F}, i = 1..v$, многочлена $M(0,y)$
4 **for** $j \leftarrow 1$ **to** v
5 **do** $\Phi[i] \leftarrow \gamma_j$
6 **if** $i = k-1$
7 **then** PRINT$(\sum_{j=0}^{i} \Phi[j]x^j)$
8 **else** $\widehat{M}(x,y) \leftarrow M(x, y+\gamma_j)$
9 $\tilde{M}(x,y) \leftarrow \widehat{M}(x,xy)$
10 RECONSTRUCT$(\tilde{M}(x,y),k,i+1,\Phi)$

Рис. 1.5. Поиск $f(x) : Q(x,f(x)) = 0$

Ли-О'Салливана [105], который будет рассмотрен в главе 5.1.1.1. Предложенный в [26] алгоритм позволяет решить эту задачу со сложностью $O((rn/k)^{O(1)}M(n)\log n)$, где $M(n)$ — сложность перемножения многочленов степени $n-1$. Однако соответствующие константы крайне велики, что делает данный алгоритм труднопримени́мым на практике.

1.5.4.2. Поиск функциональных корней

Одним из наиболее эффективных методов поиска многочленов $f(x) : Q(x,f(x)) = 0, \deg f(x) < k$, является алгоритм Рота-Рукенштейн [148]. Он основан на том, что если $Q(x,f(x))$ обращается в ноль как многочлен, то справедливо и $Q(0,f(0)) = 0$. Это дает возможность найти свободный член $f(x)$, воспользовавшись любым методом поиска корней многочленов от одной переменной, например [64]. Оставшиеся коэффициенты могут быть найдены рекурсивно, как показано на рис. 1.5. Сложность этого алгоритма составляет $O((\rho\log^2\rho)k(n+\rho\log|\mathbb{F}|))$.

Следует отметить, что этот алгоритм может возвратить некоторые паразитные решения, т.е. "усеченные" многочлены $f(x) : Q(x,f(x)) = 0, \deg f(x) \geq k$.

1.5.4.3. Перекодирование

Преобразование перекодирования [99] позволяет значительно снизить размерность задач, возникающих на различных этапах метода Гурусвами-Судана, и уменьшить тем самым

их сложность.

Т.к. коды Рида-Соломона являются линейными, вычитание из принятой последовательности любого кодового слова не влияет на результат декодирования, если последний рассматривать как список возможных векторов ошибки. Коды Рида-Соломона являются МДР-кодами, поэтому любые k позиций в кодовом слове образуют информационную совокупность. Например, некоторое кодовое слово может быть восстановлено по первым k символам принятой последовательности как $(r(x_0), \ldots, r(x_{n-1}))$, где $r(x) = \sum\limits_{i=0}^{k-1} y_i \prod\limits_{\substack{j=0 \\ j \neq i}}^{k-1} \frac{x - x_j}{x_i - x_j}$. Величины $x_i, i = 0..k-1$, будем называть локаторами перекодирования. Тогда можно построить модифицированную последовательность $y'_i = y_i - r(x_i), i = 0..n-1$. Заметим, что первые k символов этой последовательности всегда равны нулю. Это означает, что многочлен $Q'(x, y)$, имеющий точки (x_i, y'_i) в качестве корней кратности r, имеет вид $Q'(x, y) = \sum_{j=0}^{\rho} q'_j(x)\psi^{\max(r-j,0)}(x)y^j$, где $\psi(x) = \prod_{j=0}^{k-1}(x - x_j)$. Выполняя замену переменных $y = z\psi(x)$, получим $Q''(x, z) = \sum_{j=0}^{\rho} q'_j(x)\psi^{\max(r,j)}(x)z^j = \psi^r(x)\sum_{j=0}^{\rho} q'_j(x)\psi^{\max(0,j-r)}(x)z^j = \psi^r(x)Q'''(x, z)$. Многочлен $Q'''(x, z)$ имеет корни $\left(x_i, \frac{y_i - r(x_i)}{\psi(x_i)}\right), i = k..n-1$, кратности r. Таким образом, $Q(x, y) = \psi^r(x)Q'''(x, \frac{y - r(x)}{\psi(x)})$. Многочлен $Q'''(x, z)$ может быть получен непосредственно как решение соответствующей системы линейных однородных алгебраических уравнений, причем размерность этой системы оказывается существенно меньше по сравнению с аналогичной системой, возникающей в оригинальном алгоритме Гурусвами-Судана. Для решения этой системы уравнений может быть использован ИИА.

Уравнение $Q(x, f(x)) = 0$, задающее все решения исходной задачи списочного декодирования, в терминах модифицированного многочлена может быть переписано как $Q'''\left(x, \frac{f(x) - r(x)}{\psi(x)}\right) = 0$. Пусть $f(x) - r(x) = \pi(x)\omega(x)$, где

$$\pi(x) = \prod_{\substack{i=0 \\ f(x_i)=r(x_i)}}^{k-1} (x - x_i).$$

Тогда все решения задачи списочного декодирования могут быть также получены из уравнения

$$Q'''\left(x, \frac{\omega(x)}{\sigma(x)}\right) = 0,$$

где $\sigma(x) = \frac{\psi(x)}{\pi(x)}$ — многочлен, имеющий корень x_i тогда и только тогда, когда соответствующее кодовое слово отличается от принятой последовательности y_i в i-ом символе, причем x_i является одним из локаторов перекодирования.

1.5.5. Метод By списочного декодирования кодов Рида-Соломона

В работе [193] был предложен схожий подход к списочному декодированию, который, однако, требует использования существенно меньших кратностей корней, что автоматически приводит к снижению вычислительной сложности по сравнению с методом Гурусвами-Судана.

В данном разделе для простоты будем предполагать, что $x_j = \alpha^{j-1}$.

Одним из стандартных способов декодирования с исправлением числа ошибок, превышающего половину конструктивного расстояния кодов БЧХ, является метод аналитического продолжения [44]. Он состоит в подборе неизвестных компонентов вектора синдрома $S_{d-1}^{(j)}, S_{d-2}^{(j)}, \ldots$, которые позволили бы выполнить дополнительные итерации алгоритма Берлекэмпа-Месси и построить многочлен локаторов ошибок $\Lambda^{(j)}(x)$, удовлетворяющий (1.28) и соответствующий j-ому решению задачи списочного декодирования. Заметим, что этот многочлен строится как

$$\Lambda^{(j)}(x) = a^{(j)}(x)\Lambda(x) + b^{(j)}(x)xB(x), \tag{1.32}$$

где $\Lambda(x)$ — многочлен связей, построенный исходным алгоритмом Берлекэмпа-Месси (см. рис. 1.3), и $B(x)$ — последний сохраненный им многочлен $B(x)$. Основная идея алгоритма By состоит в том, что необходимо рассмотреть только такие $a^{(j)}(x), b^{(j)}(x)$, которые приводят к допустимому многочлену $\Lambda^{(j)}(x)$, число корней которого в $\mathbb{F}$ должно совпадать с его степенью. В работе [193] был предложен подход к поиску таких многочленов $a^{(j)}(x), b^{(j)}(x)$, основанный на построении многочлена $Q(x, y)$, имеющего корни $(\alpha^i, -\frac{\Lambda(\alpha^{-i})}{\alpha^{-i}B(\alpha^{-i})})$ некоторой кратности r. Однако, как будет показано ниже, приведенная в [193] оценка для значения r, требуемого для обеспечения заданного радиуса декодирования, сильно завышена, что приводит к чрезмерным вычислительным затратам. Кроме того, этот подход может потребовать проведения интерполяции по бесконечно удаленным точкам, что вызывает затруднения при использовании известных быстрых алгоритмов двумерной интерполяции.

1.6. Полиномиальные и мономиальные коды

Многие хорошие и часто используемые на практике корректирующие коды определяются как множества векторов значений многочленов в различных точках. Рассмотрим конечное поле $\mathbb{F}_{q^{ms}}$ и определим последовательность различных элементов $a_i = (a_{i,0}, \ldots, a_{i,m-1}), a_{i,j} \in \mathbb{F}_{q^s}$, некоторой его мультипликативной подгруппы порядка $n = \frac{q^{ms}-1}{b}$. В [94] полиномиальный

код был определен как

$$C_{m,s,\mu,b,q} = \left\{ \mathbf{v}(f) \middle| f = \sum_{\substack{i_0,\ldots,i_{m-1} \\ \sum_{s=0}^{m-1} i_s = jb}} f_{i_0,\ldots,i_{m-1}} X_0^{i_0} \cdots X_{m-1}^{i_{m-1}}, 0 \le j \le \mu, f_{i_0,\ldots,i_{m-1}} \in \mathbb{F}_{q^s} \right\} \cap \mathbb{F}_q^n, \tag{1.33}$$

где $\mathbf{v}(f) = (f(a_0), \ldots, f(a_{n-1}))$ — вектор значений многочлена $f = f(X_0, \ldots, X_{m-1})$ в точках $a_i \in \mathbb{F}_{q^s}^m$. В частности, при $m = 1, \mu = n - d$ получим код БЧХ в узком смысле длины n с конструктивным расстоянием d. Полагая дополнительно $s = 1$, получим код Рида-Соломона. Полагая $s = b = 1$, получим обобщенный выколотый код Рида-Маллера порядка μ длины $q^m - 1$. Однако за последние годы были введены классы корректирующих кодов, также основанные на вычислении значений многочленов в различных точках, но формально не подпадающие под определение (1.33).

В [27] было введено понятие мономиальных кодов. Пусть $\mathcal{M}_m$ — множество мономов от переменных $X_0, \ldots, X_{m-1}$. Рассмотрим некоторое его подмножество $I \subset \mathcal{M}_m$ и определим мономиальный код длины $n = q^{sm}$ как

$$C(I) = \left\{ \mathbf{V}(f) | f = \sum_i f_i X^{(i)}, f_i \in \mathbb{F}_{q^s} \right\}, \tag{1.34}$$

где $X^{(i)}$ — i-ый моном из множества I, $\mathbf{V}(f) = (f(\alpha_0), \ldots, f(\alpha_{q^{ms}-1}))$, α_j — j-ый элемент векторного пространства $\mathbb{F}_{q^s}^m$. Выбирая $m = 1$, $I = \{x^i | 0 \le i < k\}$, получим код Рида-Соломона. Выбирая $q = 2, s = 1, I = \left\{ \prod_{i=0}^{m-1} x_i^{t_i} | 0 \le t_i \le 1, \sum_i t_i \le r \right\}$, получим код Рида-Маллера порядка r. Выбирая $s = 1, I = \left\{ \prod_{i=0}^{m-1} x_i^{q-1-j_i} | \sum_{i=0}^{m-1} j_i q^{m-1-i} \in \mathcal{F} \right\}$, получим полярный код с $q \times q$ ядром Рида-Соломона (при $q = 2$ — ядром Арикана) и множеством номеров замороженных символов $\mathcal{F}$.

1.7. Выводы

Разработка перспективных систем передачи информации и хранения данных требует создания методов помехоустойчивого кодирования, обладающих простыми процедурами построения, кодирования и декодирования, а также высокой корректирующей способностью.

Полярные коды обладают первыми тремя качествами и способны достигать предела Шеннона, но демонстрируют неудовлетворительную корректирующую способность на практически значимых длинах. Для полярных кодов с ядрами, отличными от ядра Арикана, отсутствуют эффективные алгоритмы расчета вероятностей входных символов ядра (1.7). Кроме того, декодирование полярных кодов даже с ядром Арикана почти по максимуму правдоподобия требует применения достаточно сложного списочного алгоритма, в котором,

однако, существует потенциальная возможность существенного упрощения. Следует также отметить, что полярные коды являются частным случаем мономиальных кодов, структура которых близка к структуре полиномиальных кодов, многие из которых обладают хорошими дистантными свойствами. Это позволяет надеяться на создание кодовой конструкции, объединяющей преимущества полиномиальных и полярных кодов, а также расширение области применимости алгоритмов декодирования полярных кодов на иные кодовые конструкции.

Во многих приложениях (например, системах хранения данных, стеганографии) большое значение имеет гарантированная способность кода исправлять ошибки и/или стирания. В этом смысле оптимальными являются коды Рида-Соломона. Несмотря на долгую историю их исследований, существующие методы их систематического кодирования и списочного декодирования остаются чрезмерно сложными. Кроме того, даже при использовании списочных методов не удается реализовать декодирование кодов Рида-Соломона по максимуму правдоподобия.

В связи с этим в работе далее будут рассмотрены следующие задачи:

1. Разработка кодовой конструкции, обобщающей полиномиальные и мономиальные коды.
2. Построение кодов (полярных подкодов), основанных на предложенном обобщении, допускающих декодирование с помощью простых алгоритмов, предложенных для случая полярных кодов.
3. Построение упрощенных алгоритмов декодирования полярных и иных кодов (полярных подкодов, кодов БЧХ, Рида-Соломона, Голея), получаемых из предложенной обобщенной конструкции.
4. Разработка быстрых алгоритмов списочного декодирования кодов Рида-Соломона.
5. Разработка быстрого алгоритма систематического кодирования кодов Рида-Соломона.

Глава 2

Многочленные коды

В данной главе введен новый способ представления линейных блоковых кодов, применение которого позволяет в некоторых случаях использовать для их декодирования эффективные алгоритмы, разработанные первоначально для полярных кодов. Предлагаемый подход является обобщением рассмотренных ранее конструкций полиномиальных и мономиальных кодов.

В разделе 2.1 представлены предлагаемые обобщения. Их связь с конструкцией полярных кодов описана в разделе 2.2. Анализ вероятности ошибки декодирования многочленных кодов с использованием алгоритмов декодирования полярных кодов приведен в разделе 2.3. В разделах 2.4, 2.6 и 2.5 описано применение предложенного подхода, соответственно, к расширенным примитивным кодам БЧХ и Рида-Соломона в узком смысле, а также расширенному коду Голея.

2.1. Некоторые обобщения

2.1.1. Многочленные коды

Пусть $\mathcal{P}_m = \mathbb{F}_q[X_0, \dots, X_{m-1}]$ — множество многочленов, степень которых относительно каждой из переменных меньше некоторого числа l. Для произвольного множества $\mathcal{E} \subset \mathbb{F}_q^m$ и многочлена $f \in \mathcal{P}_m$ определим функцию

$$v(f, \mathcal{E}) = (f(e_0), \dots, f(e_{|\mathcal{E}|-1})),$$

результатом которой является вектор значений многочлена f в различных точках $e_i \in \mathcal{E}$. Эта функция может быть обобщена на случай вектора многочленов следующим образом:

$$\mathcal{V}\left((f^{(0)}, \dots, f^{(z-1)}), (\mathcal{E}_0, \dots, \mathcal{E}_{z-1})\right) = (v(f^{(0)}, \mathcal{E}_0), \dots, v(f^{(z-1)}, \mathcal{E}_{z-1})),$$

где $\mathcal{E}_i \subset \mathbb{F}_q^{m_i}$, $f^{(i)} \in \mathcal{P}_{m_i}$.

Определение 5. *Рассмотрим конечное множество $\mathcal{P} \subset \mathcal{P}_{m_0} \times \dots \times \mathcal{P}_{m_{z-1}}$ линейно независимых над $\mathbb{F}_q$ векторов многочленов. Пусть E — некоторое l-элементное подмножество $\mathbb{F}_q$. Многочленным кодом длины $n = \sum_{s=0}^{z-1} l^{m_s}$ над $\mathbb{F}_q$ называется*

$$C(\mathcal{P}) = \left\{ \mathcal{V}(f, (E^{m_0}, \dots, E^{m_{z-1}})) \,\middle|\, f = \sum_i f_i P^{(i)}, f_i \in \mathbb{F}_p \right\}, \mathbb{F}_p \subset \mathbb{F}_q$$

где $P^{(i)} = (p_0^{(i)}(X_0, \dots, X_{m_0-1}), \dots, p_{z-1}^{(i)}(X_0, \dots, X_{m_{z-1}-1}))$ — i-ый многочлен из множества $\mathcal{P}$.

К классу многочленных кодов можно отнести многие классические корректирующие коды. В частности, при $z = 1, E = \mathbb{F}_q, q = p^\mu$, получим:

- Код Рида-Соломона: $m_0 = 1, \mathcal{P} = \{X^0, X^1, \ldots, X^{k-1}\}$
- Код Рида-Маллера $RM(r, m_0)$: $\mathcal{P} = \left\{\prod_{i=0}^{m_0-1} X_i^{v_i} | \sum_{i=0}^{m_0-1} v_i \le r\right\}$
- Полярный код: $\mathcal{P} = \left\{\prod_{i=0}^{m_0-1} X_i^{1-v_i} | \sum_{i=0}^{m_0-1} v_i 2^{m_0-1-i} \notin \mathcal{F}\right\}$
- Код БЧХ: $\mathcal{P} = \left\{\mathrm{Tr}_{\mu_t}(\beta_t^i X^{n-1-t}) | 0 \le i < \mu_t, t \ge d, \beta_t^{p^{\mu_t}-1} = 1\right\}$, где $\mathrm{Tr}_\mu(x) = \sum_{i=0}^{\mu-1} x^{p^i}$, $\mu_t | \mu$.

Заметим, что вычисление значений многочлена от одной переменной в различных точках соответствует умножению вектора его коэффициентов на матрицу Вандермонда (ядро Рида-Соломона F_l). Вычисление значений многочлена от m_0 переменных соответствует умножению вектора его коэффициентов на матрицу $F_l^{\otimes m}$.

Таким образом, кодирование в (n, k) многочленном коде может быть выполнено как

$$c_0^{n-1} = xW \underbrace{\mathrm{diag}(A_{m_0}, \ldots, A_{m_{z-1}})}_{A},$$

где W — $k \times n$ *матрица прекодирования*, i-ая строка которой содержит коэффициенты многочлена $\mathcal{P}^{(i)}$, A — матрица смешанного поляризующего преобразования, и A_{m_j} — матрица поляризующего преобразования с ядром Рида-Соломона $F_l = \begin{pmatrix} \alpha_{l-1}^{l-1} & \alpha_{l-2}^{l-1} & \ldots & \alpha_0^{l-1} \\ \alpha_{l-1}^{l-2} & \alpha_{l-2}^{l-2} & \ldots & \alpha_0^{l-2} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{l-1}^0 & \alpha_{l-2}^0 & \ldots & \alpha_0^0 \end{pmatrix}$.

В силу обратимости матрицы A такое представление может быть найдено для произвольного линейного блокового кода из уравнения $WA = G$, где G — порождающая матрица. Это позволяет представить рассматриваемый код в виде, аналогичном полярному коду, и воспользоваться алгоритмами декодирования, разработанными для полярных кодов. Представляет интерес поиск ответов на следующие вопросы:

1. Какие известные коды допускают эффективное (вычислительно простое с достаточно хорошей корректирующей способностью) декодирование с помощью метода последовательного исключения и его обобщений?
2. Как построить многочленные коды, которые допускали бы эффективное декодирование с помощью метода последовательного исключения и его обобщений?

Первому вопросу посвящена оставшаяся часть данной главы, в то время как ответ на второй вопрос дан в главе 3.

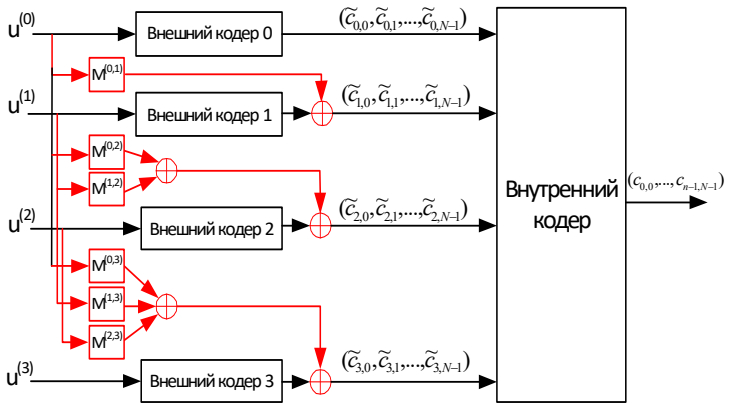


Рис. 2.1. Кодер обобщенного каскадного кода с перекрестными связями

2.1.2. Обобщенные каскадные коды с перекрестными связями

В данном разделе предлагается расширение концепции обобщенных каскадных кодов, называемое обобщенными каскадными кодами с перекрестными связями (ОККПС). Это расширение позволяет использовать для декодирования широкого класса линейных блоковых кодов методы (в частности, многошаговое декодирование), разработанные для декодирования ОКК и многоуровневых кодов. Такой подход, однако, не гарантирует хорошую (и даже сравнимую с другими известными методами декодирования) корректирующую способность в общем случае. В главе 3 будут предложены методы построения ОККПС, позволяющие получить весьма хорошую корректирующую способность при использовании списочных обобщений многошагового декодера.

Структура кодера ОККПС повторяет структуру кодера ОКК. Единственным отличием является то, что в случае ОККПС кодирование векторов $u^{(i)} \in \mathbb{F}_q^{K_i}$ осуществляется не во внешнем коде $\mathbb{C}_i$, как в классических ОКК, но в его смежном классе $\mathbb{C}_i + \left(\sum_{s=0}^{i-1} u^{(i)} M^{(s,i)}\right)$, где $M^{(s,i)} \in \mathbb{F}_q^{K_s \times N}$ — некоторая матрица, как показано на рис. 2.1. Таким образом, получается линейный блоковый код длины Nn и размерности $\sum_{i=0}^{n-1} K_i$. Оценить минимальное расстояние ОККПС достаточно сложно. Очевидно, что для любой пары положительных чисел η, k : $\eta > k$, из существования ОКК с параметрами (η, k, d) вытекает существование ОККПС с параметрами $(\eta, k, \geq d)$.

Для декодирования ОККПС может использоваться многошаговый алгоритм. Единственное отличие состоит в том, что вместо декодирования во внешних кодах следует производить декодирование в их смежных классах, определяемых по результатам предшествующих шагов. Это может быть выполнено с помощью любого декодера $\mathbb{C}_i$ путем соответствующей модификации ЛОПП, подаваемых на его вход.

2.1.3. Обобщенное разложение Плоткина

В качестве примера использования ОККПС, соответствующего внутренним кодам, порождаемым строками $\mathcal{G} = F_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$, рассмотрим обобщение классической конструкции Плоткина.

Теорема 1. *Любой линейный $(2n, k, d)$ код $\mathcal{C}$ имеет порождающую матрицу вида*

$$G = \begin{pmatrix} I_{k_1} & 0 & \tilde{I} \\ 0 & I_{k_2} & 0 \end{pmatrix} \begin{pmatrix} G_1 & 0 \\ G_2 & G_2 \\ G_3 & G_3 \end{pmatrix}, \tag{2.1}$$

где I_l — $l \times l$ единичная матрица, $G_i, 1 \le i \le 3$, — матрицы размерности $k_i \times n$, $k = k_1 + k_2$, и $\tilde{I}$ получается путем дописывания нулевой матрицы размерности $(k_1 - k_3) \times k_3$ к матрице I_{k_3}, причем $k_3 \le k_1$.

Доказательство. Пусть $\tilde{G} = \begin{pmatrix} G' & G'' \end{pmatrix}$, где G' и G'' — некоторые $k \times n$ матрицы, является порождающей матрицей рассматриваемого кода. Пусть $\tilde{H} = \begin{pmatrix} H' & H'' \end{pmatrix}$ — проверочная матрица этого же кода. Пусть G_2 — решение максимального ранга матричного уравнения $G_2(H' + H'')^T = 0$. С помощью операций над строками матрицы $\tilde{G}$ можно получить матрицу $G = Q\tilde{G} = \begin{pmatrix} G_5 & 0 \\ G_4 & G_3 \\ G_2 & G_2 \end{pmatrix}$, где Q — обратимая матрица, строки матриц G_3 и G_2 линейно независимы, и $k = k_2 + k_3 + k_5$. Видно, что

$$G = \begin{pmatrix} I_{k_5} & 0 & 0 & 0 \\ 0 & I_{k_3} & 0 & I_{k_3} \\ 0 & 0 & I_{k_2} & 0 \end{pmatrix} \begin{pmatrix} G_5 & 0 \\ G_4 - G_3 & 0 \\ G_2 & G_2 \\ G_3 & G_3 \end{pmatrix}. \tag{2.2}$$

Утверждение теоремы получается выбором $G_1 = \begin{pmatrix} G_5 \\ G_4 - G_3 \end{pmatrix}$. □

Еще одним способом получения G_1 является нахождение $G' + G''$ с последующим исключением линейно зависимых строк из полученной матрицы.

Классическая конструкция Плоткина соответствует случаю $k_3 = 0$. В связи с этим представление (2.1) будет называться обобщенным разложением Плоткина (ОРП) матрицы G или кода $\mathcal{C}$.

Применение ОРП к эквивалентным кодам может привести к кодам $\mathcal{C}_1, \mathcal{C}_2$ с различными размерностями и весовыми спектрами.

Пример 1. *Рассмотрим* $(16, 7, 6)$ *код РБЧХ, порождаемый*

$$G = \left(\begin{array}{cc} 10010110 & 10010110 \\ 01010101 & 01010101 \\ 00110011 & 00110011 \\ 00001111 & 00001111 \\ \hline 00101011 & 00011000 \\ 10000010 & 11011000 \\ 11111111 & 00000000 \end{array}\right).$$

Его ОРП задается матрицами $G_1 = \begin{pmatrix} 00110011 \\ 01011010 \\ 11111111 \end{pmatrix}, G_2 = \begin{pmatrix} 10010110 \\ 01010101 \\ 00110011 \\ 00001111 \end{pmatrix}, G_3 = \begin{pmatrix} 00011000 \\ 11011000 \end{pmatrix}.$

ОРП позволяет производить жесткое декодирование кода $\mathcal{C}$ следующим образом. Пусть $\mathcal{C}_i$ — код, порождаемый G_i. Рассмотрим зашумленное кодовое слово $(y'|y'') = (c'|c'') + (e'|e'')$, где $e = (e'|e'')$ — вектор ошибки. Вычислим $z = y' + y'' = (c' + c'') + (e' + e'')$. Выполняя декодирование z в $\mathcal{C}_1$, найдем информационный вектор u' и $c' + c'' = u'G_1$. В случае успешного выполнения этого шага, можно вычислить $\tilde{y}' = y' - u'(G_1 + \tilde{I}G_3)$ и $\tilde{y}'' = y'' - u'\tilde{I}G_3$, после чего попытаться продекодировать эти векторы в коде $\mathcal{C}_2$. Данный метод может быть легко адаптирован для случая мягкого декодирования.

Из (2.1) можно заметить, что $\mathcal{C}_2$ имеет минимальное расстояние $d_2 \geq d/2$. Однако d_1 может быть весьма мало. Таким образом, описанный алгоритм декодирования может оказаться неспособен исправить даже $t \leq \lfloor (d-1)/2 \rfloor$ ошибок. Эту проблему можно преодолеть с использованием списочного декодера кода $\mathcal{C}_1$, который может найти несколько наиболее вероятных векторов u', для каждого из которых можно продекодировать соответствующие $\tilde{y}', \tilde{y}''$ в коде $\mathcal{C}_2$ и выбрать кодовое слово $(c'|c'')$, ближайшее к принятой последовательности.

ОРП может применяться рекурсивно. Это приводит к кодам длины 1 и размерности не более 1, как описано ниже.

2.2. Динамически замороженные символы

Рассмотрим $(n = l^m, k, d)$ код $\mathcal{C}$ над $\mathbb{F}_q$ с проверочной матрицей H. Пусть $A_m = B_{l,m}F_l^{\otimes m}$ — матрица $n \times n$ поляризующего преобразования. Т.к. A_m обратима, любой вектор длины n может быть получен как результат $c_0^{n-1} = u_0^{n-1}A_m$ применения поляризующего преобразова-

ния к подходящему вектору u_0^{n-1}. Изучим ограничения, которые должны быть наложены на u_0^{n-1}, чтобы результат его применения принадлежал коду $\mathcal{C}$.

Эти ограничения задаются уравнением $u_0^{n-1} A_m H^T = 0$. Путем элементарных преобразований строк можно получить *матрицу ограничений* $V = QHA_m^T$, где Q — такая обратимая матрица, что последние ненулевые элементы всех строк V расположены в различных столбцах, т.е. значения $j_i = \max\{t | V_{i,t} \neq 0\}, 0 \leq i < n-k$ различны. Не ограничивая общности, будем считать, что $V_{i,j_i} = -1$. Пусть $\mathcal{F} = \{j_i | 0 \leq i < n-k\}$. Тогда получим

$$u_{j_i} = \sum_{s=0}^{j_i - 1} u_s V_{i,s}, 0 \leq i < n-k. \tag{2.3}$$

Эти уравнения могут рассматриваться как обобщение ограничений, налагаемых на замороженные символы в полярных кодах $u_{j_i} = 0, j_i \in \mathcal{F}$. Заметим, что символы $u_{j_i}, j_i \in \mathcal{F}$ могут принимать произвольные значения, которые, однако, зависят от значений символов с меньшими номерами. Таким образом, символы u_{j_i}, задаваемые (2.3), будем называть динамически замороженными [178]. Если все коэффициенты $V_{i,s}$ в правой части (2.3) являются нулевыми, будем называть соответствующие символы u_{j_i} статически замороженными.

В случае $l = 2$ описанная конструкция получается в результате рекурсивного применения ОРП к коду $\mathcal{C}$.

Пример 2. *Рассмотрим* $(16, 7, 6)$ *расширенный примитивный код БЧХ в узком смысле* $\mathcal{C}$ *над* $\mathbb{F}_2$. *Порождающий многочлен соответствующего нерасширенного кода имеет корни* α, α^3 *и их сопряженные, где* α *— примитивный корень* $x^4 + x^3 + 1$. *Пусть* $A = B_{2,4} F_2^{\otimes 4}$.

Ограничения на вектор u_0^{15}*, обеспечивающие* $u_0^{15}A_4 \in \mathcal{C}$*, имеют вид*

$$u_0^{15}\begin{pmatrix}1000000000000000\\1000000010000000\\1000100000000000\\1000100010001000\\1010000000000000\\1010000010100000\\1010101000000000\\1010101010101010\\1100000000000000\\1100000011000000\\1100110000000000\\1100110011001100\\1111000000000000\\1111000011110000\\1111111100000000\\1111111111111111\end{pmatrix}\begin{pmatrix}1 & 0 & 0\\1 & 1 & 1\\1 & \alpha & \alpha^3\\1 & 1+\alpha & (1+\alpha)^3\\1 & \alpha^2 & 1+\alpha+\alpha^2+\alpha^3\\1 & 1+\alpha^2 & 1+\alpha\\1 & \alpha(1+\alpha) & 1+\alpha^2\\1 & 1+\alpha+\alpha^2 & 1+\alpha+\alpha^2+\alpha^3\\1 & \alpha^3 & 1+\alpha^2\\1 & 1+\alpha^3 & 1+\alpha\\1 & \alpha(1+\alpha^2) & 1\\1 & 1+\alpha+\alpha^3 & 1\\1 & \alpha^2(1+\alpha) & 1+\alpha\\1 & 1+\alpha^2+\alpha^3 & \alpha^3\\1 & \alpha(1+\alpha+\alpha^2) & 1+\alpha^2\\1 & 1+\alpha+\alpha^2+\alpha^3 & \alpha^3\end{pmatrix} = 0.$$

Перемножая матрицы, разлагая их элементы в стандартном базисе и применяя элементарные линейные преобразования, получим

$$u_0^{15}\begin{pmatrix}0000000000101000\\0001010000100000\\0000010001000000\\0000000010000000\\0001001000000000\\0000100000000000\\0010000000000000\\0100000000000000\\1000000000000000\end{pmatrix}^T = 0$$

Это означает, что $u_0 = u_1 = u_2 = u_4 = u_8 = 0$*, и* $u_6 = u_3$*,* $u_9 = u_5$*,* $u_{10} = u_3 + u_5$*,* $u_{12} = u_{10}$*. Символы* $u_3, u_5, u_7, u_{11}, u_{13}, u_{14}, u_{15}$ *являются незамороженными.*

Заметим, что любой код длины l^m может быть представлен в виде системы линейных уравнений (2.3). Это позволяет применить метод последовательного исключения и его обобщения для декодирования произвольных линейных кодов длины l^m. Т.е., декодирование мо-

жет производиться путем последовательного принятия решений

$$\widehat{u}_i = \begin{cases} \arg\max_{u_i \in \mathbb{F}_q} \mathcal{W}_m^{(i)} \left\{ \widehat{u}_0^{i-1}, u_i | y_0^{n-1} \right\}, & i \notin \mathcal{F} \\ \sum_{s=0}^{i-1} \widehat{u}_s V_{t_i,s}, & \text{иначе,} \end{cases} \tag{2.4}$$

где t_i — целое такое, что $j_{t_i} = i$. Заметим, что если $\widehat{u}_0^{i-1}$ являются истинными значениями входных символов поляризующего преобразования, вероятность ошибки P_i на каждом шаге этого алгоритма совпадает с таковой для классических полярных кодов. Таким образом, вероятность ошибки декодирования кода методом последовательного исключения может быть вычислена с помощью (1.10).

Вероятности $\mathcal{W}_n^{(i)}(u_0^i | y_0^{n-1})$ могут быть вычислены тем же способом, что и в случае классических полярных кодов, что требует $O(n \log n)$ операций. Вместе с тем, вычисление значений выражений (2.3) может увеличить общую сложность декодирования до $O(n^2)$.

Описанный подход может быть обобщен и на случай кодов с длинами n, отличными от l^m. Пусть $n = \sum_{i=0}^{s-1} l^{m_i}$, где m_i — различные натуральные числа. Введем смешанное поляризующее преобразование $A = \operatorname{diag}(B_{l,m_0} F_l^{\otimes m_0}, \ldots, B_{l,m_{s-1}} F_l^{\otimes m_{s-1}})$. Далее матрица ограничений может быть построена как $V = QHA^T$, где Q — обратимая матрица такая, что последние ненулевые элементы строк V расположены в различных столбцах. Методы декодирования таких кодов рассмотрены в разделе 4.2.

Представление линейного блокового кода с помощью матрицы ограничений эквивалентно его представлению в виде многочленного кода. Очевидно, что матрицы ограничений и прекодирования при этом связаны соотношением

$$WV^T = 0.$$

Не ограничивая общности будем считать, что первые ненулевые элементы строк матрицы W расположены в различных столбцах l_j, причем все столбцы l_j матрицы W имеют вес 1. Заметим, что

$$\mathcal{N} = \{l_j | 0 \le j < k\} = \{0, \ldots, n-1\} \setminus \mathcal{F}.$$

В некоторых случаях будет удобно использовать следующее эквивалентное описание метода ПИ:

1. Пусть $\hat{u}_i \leftarrow 0, i \in \mathcal{F}$
2. Для i с 0 по $n-1$, где $i \notin \mathcal{F}$:
   - Пусть $\hat{u}_i \leftarrow \arg\max_{u_i \in \mathbb{F}_q} \mathcal{W}_m^{(i)}(y_0^{n-1}, \hat{u}_0^{i-1} | u_i)$

- Пусть $\hat{u}_j \leftarrow \hat{u}_j + \hat{u}_i, j > i, W_{t_i,j} = 1$, где t_i — номер строки матрицы W, содержащей первый ненулевой элемент в столбце i.

Множество $\mathcal{F}$ номеров замороженных символов произвольного линейного кода в общем случае может содержать не только номера i подканалов с высокой вероятностью ошибки P_i. В связи с этим, вероятность ошибки декодирования методом последовательного исключения, задаваемая (1.10), для многих линейных кодов оказывается существенно выше по сравнению с иными известными алгоритмами декодирования. Однако существенно лучшие результаты могут быть получены при использовании списочного и последовательного алгоритмов декодирования. Вместе с тем, размер списка (т.е. сложность) в этих алгоритмах, требуемый для обеспечения декодирования по максимуму правдоподобия, в общем случае растет экспоненциально с увеличением размерности кода. Действительно, размер списка должен быть таким, чтобы обеспечить возможность исправления перебором наиболее вероятных конфигураций ошибок в незамороженных символах. Число таких конфигураций растет экспоненциально с увеличением числа недостаточно надежных незамороженных символов. Вместе с тем, точный анализ поведения списочного алгоритма Тала-Варди и его аналогов остается открытой задачей.

Представление линейного блокового кода длины $n = l^m$ с помощью системы ограничений динамического замораживания эквивалентно его представлению в виде ОККПС, где внутренние коды порождаются строками $A_{m-s}, 0 < s \leq m$, а внешние коды являются полярными кодами с множествами номеров замороженных символов, задаваемых (1.19), где $\mathcal{F} = \{j_i | 0 \leq i < l^m - k\}$. При этом матрицы $M^{(i,j)}$ могут быть найдены из матрицы ограничений V.

2.3. Анализ вероятности ошибки декодирования

Оценим вероятность ошибки декодирования многочленного кода длины 2^m над $\mathbb{F}_2$, представленного в виде системы ограничений динамического замораживания для случая ядра Арикана. При использовании метода последовательного исключения она удовлетворяет

$$P_{SC} \leq \sum_{i \notin \mathcal{F}} P_i \leq \sum_{i \notin \mathcal{F}} Z_{m,i}.$$

Ограничимся случаем двоичного стирающего канала. В этом случае параметры Бхаттачарьи битовых подканалов удовлетворяют $Z_{m,i} = A_i Z_{0,0}^{2^{\mathrm{wt}(i)}} + o(Z_{0,0}^{2^{\mathrm{wt}(i)}})$, где A_i — некоторые коэффициенты, которые могут быть найдены из (1.16)–(1.17). Таким образом, получим

$P_{SC} \le AZ_{0,0}^{\delta} + o(Z_{0,0}^{\delta})$, где $\delta = 2^w, w = \min_{i \notin \mathcal{F}} \mathrm{wt}(i)$ и

$$A = \sum_{\substack{i \notin \mathcal{F} \\ \mathrm{wt}(i)=w}} A_i.$$

Анализ корректирующей способности списочного метода Тала-Варди до настоящего времени остается открытой проблемой. В случае двоичного стирающего канала вероятность ошибки этого метода с размером списка $L = 2^k$ не превосходит вероятности ошибки следующего суррогатного декодера [121]. Будем считать, что имеется возможность получить не более k подсказок относительно значений тех символов u_i, для которых на выходе канала $W_m^{(i)}$ при декодировании появилось стирание. В этом случае вероятность ошибки декодирования кода длины N со скоростью R удовлетворяет

$$P_e(N, R, Z_{0,0}, k) \ge \left(\frac{3}{16}\right)^{2^k-1} (P_e(N, R, Z_{0,0}, 0))^{2^k},$$

где $P_e(N, R, Z_{0,0}, 0) = P_{SC}$ [121]. Таким образом, получим, что

$$P_e(N, R, Z_{0,0}, k) \ge \left(\frac{3}{16}\right)^{2^k-1} (AZ_{0,0}^{\delta})^{2^k} + o(Z_{0,0}^{\delta 2^k}).$$

Отсюда следует, что декодирование с помощью (списочного) алгоритма последовательного исключения линейных блоковых кодов с использованием предложенного представления может быть эффективным только для кодов, характеризующихся достаточно большими значениями δ (т.е. представимых в виде векторов значений многочленов достаточно малой степени) и малыми значениями A.

2.4. Расширенные коды БЧХ

2.4.1. Структура ограничений динамического замораживания

Рассмотрим более подробно структуру ограничений динамического замораживания для двоичных расширенных примитивных кодов БЧХ в узком смысле (РБЧХ) в случае ядра Арикана.

Определение 6. *$(n = q^m, k, \ge d)$ расширенный примитивный код БЧХ в узком смысле (РБЧХ) — множество векторов $c_0^{n-1} \in \mathbb{F}_q^n$ таких, что*

$$\sum_{i=0}^{n-1} c_i x_i^j = 0, 0 \le j < d-1, \tag{2.5}$$

где $(x_0, \ldots, x_{n-1})$ — вектор различных элементов $\mathbb{F}_{q^m}$, называемых локаторами кода.

Полагая $x_0 = 0, x_i = \alpha^{i-1}, 1 \le i < n$, получим расширенный циклический код с порождающим многочленом $g(x) = LCM(M_1(x), \dots, M_{d-2}(x))$, где $M_i(x)$ — минимальный многочлен α^i, и α — примитивный элемент $\mathbb{F}_{q^m}$. Однако в данной работе будет удобно считать, что x_i расположены в стандартном битовом порядке, где $x_i = \sum_{j=0}^{m-1} X_{i,j}\beta_j$, $i = \sum_{j=0}^{m-1} X_{i,j}q^j, X_{i,j} \in \{0, \dots, q-1\}$, и $\beta_0, \dots, \beta_{m-1}$ — некоторый базис $\mathbb{F}_{q^m}$. Здесь $X_{i,j}$ обозначает как целое число, так и элемент $\mathbb{F}_q$. Далее в этом разделе ограничимся рассмотрением двоичных кодов.

В работах [100, 93, 51] было показано, что выколотый двоичный код Рида-Маллера порядка r и длины $2^m - 1$ эквивалентен циклическому коду с порождающим многочленом $g(x)$, имеющим корни $\alpha^i : 1 \le \mathrm{wt}(i) < m-r, 1 \le i \le 2^m - 2$, где α — примитивный элемент $\mathbb{F}_{2^m}$, и $\mathrm{wt}(i)$ — число ненулевых цифр в двоичном разложении i. Кроме того, в [196] было показано, что код РБЧХ $\mathcal{C}'$ длины 2^m с конструктивным расстоянием $d \ge \delta(r,m)+3$ является подкодом кода Рида-Маллера порядка $m-r-1$, где

$$\delta(r,m) = \max_{i:\mathrm{wt}(i)=r} \min\left\{i2^j \bmod (2^m-1) | 0 \le j < m\right\}.$$

Рекурсивное выражение для $\delta(r,m)$ приведено в [196]. Код Рида-Маллера порядка $m-r-1$ можно рассматривать как полярный код с ядром Арикана с множеством номеров замороженных символов $\mathcal{F}'' = \{i | \mathrm{wt}(i) \le r\}$. Таким образом, множество номеров динамически замороженных символов $\mathcal{F}'$ кода РБЧХ включает $\mathcal{F}''$. Из (1.15)–(1.16) можно увидеть, что $Z_{n,i} = O(Z_{1,0}^{2^{\mathrm{wt}(i)}})$. Следовательно, множество номеров динамически замороженных символов кода РБЧХ включает все номера таких подканалов, что их параметры Бхаттачарьи (и вероятности ошибки $P_i \le \frac{1}{2}Z_{n,i}$) убывают достаточно медленно с уменьшением параметра Бхаттачарьи $Z_{1,0}$ исходного канала. Большинство этих подканалов характеризуются высокой вероятностью ошибки P_i.

Обозначим

$$C_t = \left\{t2^i \bmod 2^m - 1 | 0 \le i < m_t, t2^{m_t} \equiv t \bmod 2^m - 1\right\}$$

циклотомический класс над $\mathbb{F}_2$, порождаемый t. Пусть $\mathcal{Q}$ — множество наименьших представителей циклотомических классов по модулю $2^m - 1$. Видно, что все элементы циклотомического класса имеют одинаковый вес. Таким образом,

$$\sum_{\substack{s \in \mathcal{Q} \\ \mathrm{wt}(s)=r}} m_s = \binom{m}{r}.$$

Теорема 2. *Рассмотрим $(2^m, k, d)$ код РБЧХ над $\mathbb{F}_2$ и поляризующее преобразование $A_m = B_{2,m}F_2^{\otimes m}$. Пусть $S = \{i \in \mathcal{Q} | 0 \le i < d-1\}$. Пусть N_t — число номеров i ДЗС u_i этого кода таких, что $\mathrm{wt}(i) = t$. Тогда $N_t = \sum_{s \in S_t} m_s$, где $S_t = \{s \in S | \mathrm{wt}(s) = t\}$, и m_s — мощность циклотомического класса, порождаемого s.*

Доказательство. Рассмотрим проверочное уравнение (2.5). Пусть $x_i = \sum_{s=0}^{m-1} X_{i,s}\beta_s, X_{i,s} \in \{0,1\}$. Тогда

$$x_i^j = \left(\sum_{s=0}^{m-1} X_{i,s}\beta_s\right)^j = \left(\sum_{s=0}^{m-1} X_{i,s}\beta_s\right)^{\sum_{t=0}^{m-1} j_t 2^t} = \prod_{t=0}^{m-1}\left(\sum_{s=0}^{m-1} X_{i,s}\beta_s^{2^t}\right)^{j_t} = \sum_{\substack{\mathrm{wt}(e_0^{m-1})\le \mathrm{wt}(j)\\ e_s\in\{0,1\}}} v_{j;e_0^{m-1}} \prod_{s=0}^{m-1} X_{i,s}^{e_s},$$

где $v_{j;e_0^{m-1}} \in \mathbb{F}_{2^{m_j}}$ — некоторые коэффициенты. Таким образом, любое кодовое слов c_0^{n-1} удовлетворяет

$$0 = \sum_{\substack{\mathrm{wt}(e_0^{m-1})\le \mathrm{wt}(j)\\ e_s\in\{0,1\}}} v_{j;e_0^{m-1}} \sum_{i=0}^{n-1} c_i \prod_{s=0}^{m-1} X_{i,s}^{e_s}, j \in \tilde{S},$$

где $\tilde{S} = \{j2^l | j \in S, 0 \le l < m_j, j2^{m_j} \equiv j \bmod 2^m - 1\}$.

Можно заметить, что i-ая строка матрицы $A_m = B_{2,m}F_2^{\otimes m}$ представляет собой последовательность значений одночленов $X^{(a)} = \prod_{s=0}^{m-1} X_s^{a_{m-1-s}}, a_s \in \{0,1\}$ в точке $(X_{i,0}, \ldots, X_{i,m-1}) \in \mathbb{F}_2^m$. Следовательно, $u_{e'} = \sum_{i=0}^{n-1} c_i \prod_{s=0}^{m-1} X_{i,s}^{e_s}$ равно элементу с номером e' входного вектора поляризующего преобразования, где $e' = \sum_{s=0}^{m-1} e_s 2^{m-1-s}$, $e_s \in \{0,1\}$, и $u_0^{n-1} = c_0^{n-1}A_m$, так что

$$0 = \sum_{\mathrm{wt}(e_0^{m-1})\le \mathrm{wt}(j)} v_{j;e_0^{m-1}} u_{e'}, j \in \tilde{S}$$

Каждое такое уравнение эквивалентно m_j уравнениям с коэффициентами из $\mathbb{F}_2$. Заметим, что имеется $\widetilde{M}_t = \sum_{i=t}^{\rho}\sum_{s\in S_i} m_s$ уравнений, которые включают символы $u_{e'} : t \le \mathrm{wt}(e') \le \rho$, где $\rho = \max_{0\le j<d-1}\mathrm{wt}(j)$. Следовательно, число $\widehat{M}_t = \sum_{i=t}^{\rho} N_t$ ДЗС $u_i : t \le \mathrm{wt}(i) \le \rho$, ограничено сверху $\widetilde{M}_t$. При этом можно считать, что $\widehat{M}_0 = \widetilde{M}_0$.

Равенство $N_0 = 1 = m_0$ выполняется для любого РБЧХ кода с $d \ge 2$. Предположим, что $N_t = \sum_{s\in S_t} m_s$ для всех $t < t_0$, так что $\widehat{M}_{t_0} = \widetilde{M}_{t_0}$. Т.к. $\widehat{M}_{t_0+1} = \widehat{M}_{t_0} - N_{t_0} \le \widetilde{M}_{t_0+1} = \widetilde{M}_{t_0} - \sum_{s\in S_{t_0}} m_s$, справедливо неравенство $N_{t_0} \ge \sum_{s\in S_{t_0}} m_s$. Предположим, что оно является строгим.

Любое кодовое слово рассматриваемого кода РБЧХ может быть представлено как вектор значений

$$f(x) = \sum_{\tau\in\mathcal{Q}\setminus S} \mathrm{Tr}_{m_\tau}(\gamma_\tau x^{n-1-\tau}) \tag{2.6}$$

в различных точках $x_i \in \mathbb{F}_{2^m}$, где $\gamma_i \in \mathbb{F}_{2^{m_i}}$, $\mathrm{Tr}_m(x) = \sum_{i=0}^{m-1} x^{2^i}$. Этот полином может быть переписан как многочлен от нескольких переменных

$$\begin{aligned} f(X_0, \ldots, X_{m-1}) &= \sum_{\tau\in\mathcal{Q}\setminus S} \mathrm{Tr}_{m_\tau}\left(\gamma_\tau \prod_{j=0}^{m-1}\left(\sum_{l=0}^{m-1}\beta_l^{2^j} X_l\right)^{1-\tau_j}\right) \\ &= \sum_{e_0,\ldots,e_{m-1}\in\{0,1\}} u_{e'} \prod_{l=0}^{m-1} X_l^{1-e_l}, \end{aligned} \tag{2.7}$$

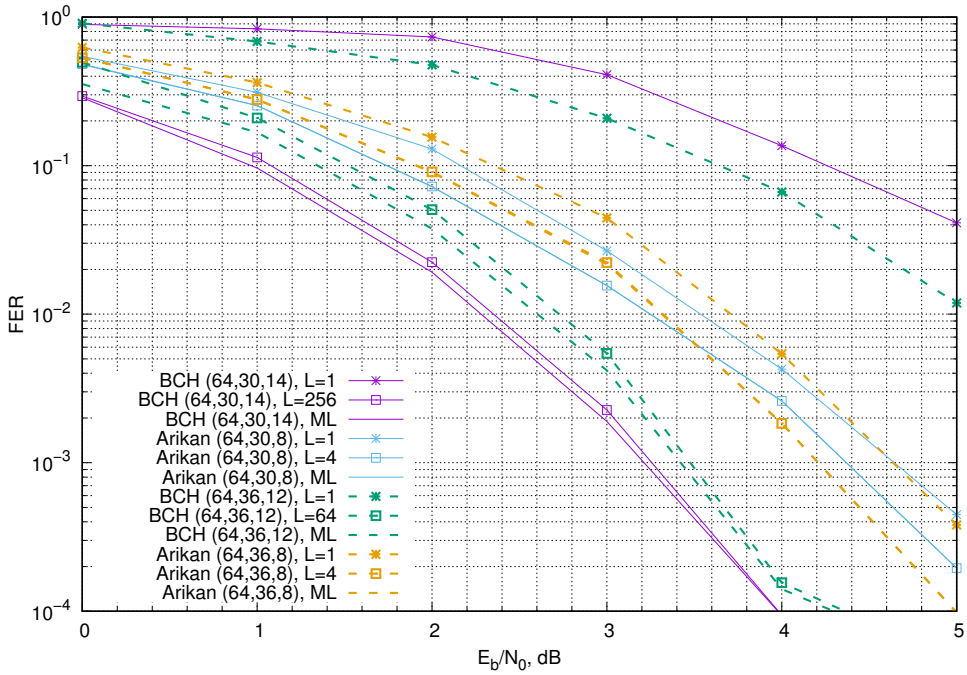


Рис. 2.2. Вероятность ошибки декодирования кодов РБЧХ списочным алгоритмом ПИ

где $e' = \sum_{l=0}^{m-1} e_l 2^l$, $u_{e'}$ — некоторые коэффициенты, $\tau = \sum_{j=0}^{m-1} \tau_j 2^j, \tau_j \in \{0,1\}$, так что $c_i = f(x_i) = f(X_{i,0}, \ldots, X_{i,m-1})$. Заметим, что строка e' матрицы A_m представляет собой последовательность значений $\prod_{l=0}^{m-1} X_l^{1-e_l}$ в различных точках $\mathbb{F}_2^m$. Следовательно, $u_{e'}$ могут рассматриваться как входные символы поляризующего преобразования.

Следовательно, множество многочленов $f(X_0, \ldots, X_{m-1})$, соответствующих рассматриваемому коду, содержит $K_{t_0} = \binom{m}{t_0} - \sum_{s \in S_{t_0}} m_s$ линейно независимых многочленов вида (2.7) степени $m - t_0$. Заметим, что формы степени $m - t_0$ этих многочленов также линейно независимы. Но это невозможно, т.к. выше было выдвинуто предположение, что существует $N_{t_0} > \sum_{s \in S_{t_0}} m_s$ ограничений на коэффициенты этих форм. Полученное ограничение доказывает теорему. □

Конкретное множество номеров ДЗС $\mathcal{F}'$ кода РБЧХ зависит от используемого базиса. Можно перебрать различные базисы $\mathbb{F}_{2^m}$ и выбрать тот, который минимизирует вероятность ошибки декодирования методом ПИ (1.10). Аналогичный подход использовался в [186, 135] для построения решеток кодов РБЧХ.

2.4.2. Численные результаты

Теорема 2 и существование надкода Рида-Маллера позволяют предположить, что метод ПИ и его аналоги могут позволить использовать метод ПИ и его аналоги для декодирования кодов РБЧХ. Однако эксперименты показывают, что приемлемое соотношение корректирую-

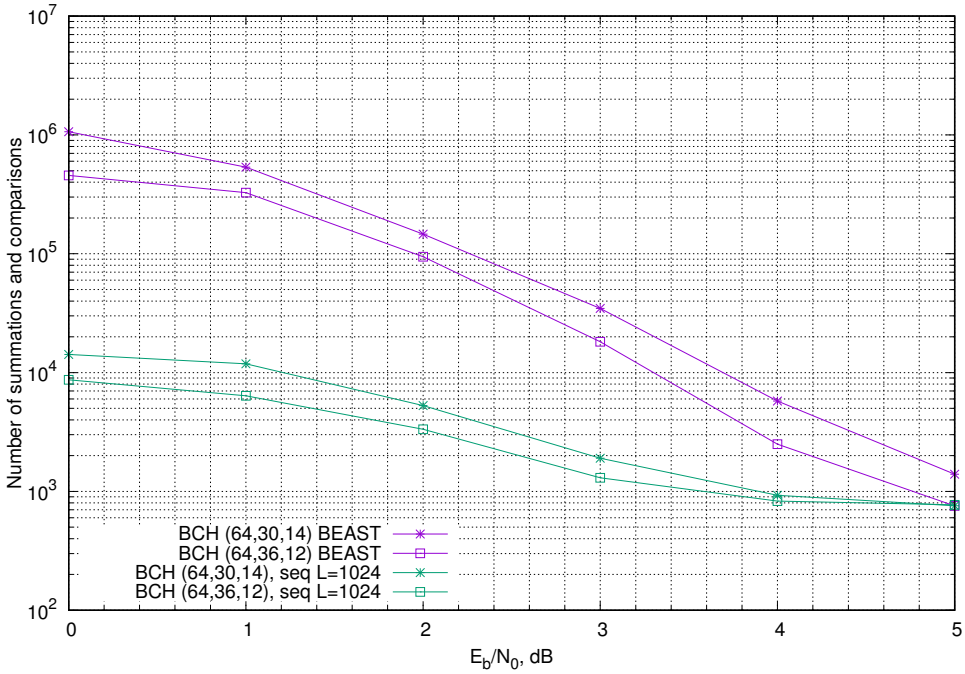


Рис. 2.3. Средняя сложность декодирования кодов РБЧХ

щей способности и сложности достигается только на малых длинах n. Рис. 2.2 иллюстрирует корректирующую способность кодов РБЧХ при использовании последовательного алгоритма ПИ с различными размерами списка L, а также при использовании декодера максимального правдоподобия. Для сравнения приведены результаты для классических полярных кодов Арикана. Можно заметить, что последние существенно превосходят коды РБЧХ в случае размера списка $L = 1$ (т.е. алгоритма ПИ). Однако большее минимальное расстояние обеспечивает значительный энергетический выигрыш кодов РБЧХ при декодировании почти по максимуму правдоподобия. Чрезвычайно большой размер списка L требуется для получения сопоставимой вероятности ошибки при декодировании списочным/стековым алгоритмом ПИ, в то время как в случае полярных кодов Арикана декодирование почти по максимуму правдоподобия обеспечивается уже при $L = 4$.

На рис. 2.3 представлено среднее число операций сложения и сравнения, выполняемых последовательным алгоритмом декодирования, описанным в разделе 4.1, при декодировании некоторых кодов РБЧХ длины 64. Для сравнения приведена средняя сложность алгоритма BEAST [38], который является высокоэффективным алгоритмом декодирования произвольных линейных блоковых кодов по максимуму правдоподобия. Видно, что предлагаемый подход обеспечивает существенное снижение средней сложности декодирования, особенно в области малых отношений сигнал/шум.

2.5. Расширенный код Голея

$(24, 12, 8)$ расширенный код Голея является квазисовершенным самодуальным кодом [18]. Одним из возможных способов его построения является конструкция Турина [34], которая состоит в следующем. Кодовые слова кода формируются как

$$c = (u + v, u + w, u + v + w), v, w \in C', u \in C'',$$

где C' — (8,4,4) расширенный код Хемминга, C'' — код, эквивалентный C, причем $C' \cap C'' = \{(0,0,0,0,0,0,0,0), (1,1,1,1,1,1,1,1)\}$. Отметим, что оба этих кода являются кодами РБЧХ. Их порождающие и проверочные матрицы имеют вид

$$G' = H' = \begin{array}{c} \begin{matrix} 0 & \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \end{matrix} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

и

$$G'' = H'' = \begin{array}{c} \begin{matrix} 0 & \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \end{matrix} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array}.$$

Столбцы этих матриц пронумерованы элементами поля $\mathbb{F}_{2^3}$. Выписывая их в стандартном битовом порядке $(0, 1, \alpha, \alpha + 1 = \alpha^3, \alpha^2, \alpha^2 + 1 = \alpha^6, \alpha^2 + \alpha = \alpha^4, \alpha^2 + \alpha + 1 = \alpha^5)$ и комбинируя матрицы в соответствии с конструкцией Турина, получим порождающую и проверочную матрицу расширенного кода Голея в виде

$$G = H = \left(\begin{array}{cccccccc|cccccccc|cccccccc}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
1&0&1&0&1&0&1&0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&1&0\\
1&0&0&1&1&0&0&1&0&0&0&0&0&0&0&0&1&0&0&1&1&0&0&1\\
1&0&0&1&0&1&1&0&0&0&0&0&0&0&0&0&1&0&0&1&0&1&1&0\\
\hline
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0\\
0&0&0&0&0&0&0&0&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1\\
0&0&0&0&0&0&0&0&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0\\
\hline
1&1&0&1&1&0&0&0&1&1&0&1&1&0&0&0&1&1&0&1&1&0&0&0\\
1&0&1&1&0&0&1&0&1&0&1&1&0&0&1&0&1&0&1&1&0&0&1&0\\
1&0&0&0&1&0&1&1&1&0&0&0&1&0&1&1&1&0&0&0&1&0&1&1\\
1&0&0&1&0&1&0&1&1&0&0&1&0&1&0&1&1&0&0&1&0&1&0&1
\end{array}\right)$$

Для того, чтобы применить методы декодирования полярных кодов к расширенному коду Голея, введем смешанное поляризующее преобразование $A = \mathrm{diag}(B_{2,4}F_2^{\otimes 4}, B_{2,3}F_2^{\otimes 3})$. Тогда можно построить матрицу ограничений вида

$$V = QHA^T = \left(\begin{array}{cccccccccccccccc|cccccccc}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0
\end{array}\right)$$

Применение полученного представления расширенного кода Голея для его декодирования будет рассмотрено в п. 4.3.

2.6. Расширенные коды Рида-Соломона

2.6.1. Структура ограничений динамического замораживания

Рассмотрим представление расширенных примитивных кодов Рида-Соломона в узком смысле над $\mathbb{F}_{2^m}$ в виде системы ограничений (2.3) для случая $A_m = B_{2,m}F_2^{\otimes m}$. Всякое кодовое слово $(n = 2^m, k, 2^m - k + 1)$ кода Рида-Соломона может быть представлено как $c = (f(x_0), f(x_1), \ldots, f(x_{n-1}))$, где $f(x) = \sum_{i=0}^{k-1} f_i x^i, f_i \in \mathbb{F}_{2^m}$. С другой стороны, для любого расширенного кода Рида-Соломона может быть построена проверочная матрица, с помощью которой можно получить его представление в виде (2.3).

Пусть $x = \sum_{i=0}^{m-1} x_i a_i, x_i \in \mathbb{F}_2$ — представление $x \in \mathbb{F}_{2^m}$ в некотором базисе $a = (a_0, \ldots, a_{m-1})$ поля $\mathbb{F}_{2^m}$, и $j = \sum_{s=0}^{m-1} j_s 2^s$ — представление числа j в двоичной системе счисления. Тогда $x^j = (\sum_{i=0}^{m-1} x_i a_i)^{\sum_{s=0}^{m-1} j_s 2^s} = \prod_{s=0}^{m-1} \left(\sum_{i=0}^{m-1} x_i a_i^{2^s}\right)^{j_s}$. Раскрывая скобки в этом выражении, получим некоторый многочлен от переменных $x_0, \ldots, x_{m-1}$ степени $\mathrm{wt}(j) = \sum_{s=0}^{m-1} j_s$, который может быть представлен как $x^j = \sum_{t \in Q_j} w_{jt} \prod_{i=0}^{m-1} x_i^{t_i}$, где Q_j — множество мультистепеней $t = (t_0, \ldots, t_{m-1})$ ненулевых членов, и $w_{jt} \in \mathbb{F}_{2^m}$ — некоторые коэффициенты. Таким образом,

$$f(x) = \sum_{j=0}^{k-1} f_j \sum_{t \in Q_j} w_{jt} \prod_{i=0}^{m-1} x_i^{t_i}. \tag{2.8}$$

Данное выражение представляет собой обобщенный полином Жегалкина $f : \mathbb{F}_2^m \to \mathbb{F}_{2^m}$. Таким образом, кодовое слово кода Рида-Соломона может быть представлено как

$$c = f\mathbf{W}(a)B_{2,m}F_2^{\otimes m}, \tag{2.9}$$

где $\mathbf{W}(a) = (w_{jt})$ и $\mathbf{W}(a)B_{2,m}F_2^{\otimes m}$ — порождающая матрица кода Рида-Соломона. Отметим, что строка $t = \sum_{i=0}^{m-1} t_{m-1-i} 2^i$ матрицы $B_{2,m}F_2^{\otimes m}$ представляет собой последовательность значений многочлена $\prod_{i=0}^{m-1} x_i^{1-t_i}$ в различных точках $(x_0, \ldots, x_{m-1})$. Соответствующая матрица ограничений $V = \mathbf{V}(a)$ удовлетворяет условию $\mathbf{W}(a)\mathbf{V}^T(a) = 0$. Следует также заметить, что матрицы $\mathbf{W}(a)$ и $\mathbf{V}(a)$ зависят от используемого базиса a.

Объединяя члены $f(x)$ с номерами j, принадлежащими одинаковым циклотомическим классам $C_s = \{s2^i | 0 \le i < m_s, s2^{m_s} \equiv s \bmod 2^m - 1\}$, получим, что

$$f(x) = \sum_{s \in S} \sum_{i=0}^{m_s - 1} f_{s2^i \bmod (2^m-1)} x^{s2^i} = \sum_{s \in S} L_s(x^s) = \sum_{s \in S} \sum_{t \in Q_s} L_s(w_{st}) \prod_{i=0}^{m-1} x_i^{t_i}, \tag{2.10}$$

где $L_s(x) = \sum_{i=0}^{m_s - 1} f_{s2^i \bmod (2^m-1)} x^{2^i}$ и S — множество представителей циклотомических классов.

Теорема 3. *$(2^m, k, 2^m - k + 1)$ расширенный код Рида-Соломона над $\mathbb{F}_{2^m}$ может быть представлен в виде системы ограничений замораживания с множеством номеров замороженных символов $\mathcal{F} = \{0, \ldots, 2^m - 1\} \setminus \{2^m - 1 - r_m(i) | 0 \le i < k\}$, где $r_m(i) = \sum_{j=0}^{m-1} i_j 2^{m-j-1}$ — целое число, полученное обращением битов числа $i = \sum_{j=0}^{m-1} i_j 2^j, i_j \in \{0, 1\}$.*

Доказательство. Докажем утверждение теоремы по индукции по числу проверочных символов $r = 2^m - k$. Для $r = 1$ утверждение теоремы очевидно. Предположим, что оно верно для случая кода с $r-1$ проверочным символом. Из (2.10) видно, что при уменьшении размерности кода на 1, т.е. добавлении одного проверочного символа, ранг матрицы $\mathbf{W}(a)$ уменьшается на 1, при этом линейно зависимыми оказываются столбцы с номерами, вес которых равен $m - \mathrm{wt}(k) = \mathrm{wt}(2^m - 1 - k), k = 2^m - r$. Последний такой столбец, не входящих в множество замороженных символов для кода большей размерности, имеет номер $2^m - 1 - r_m(k)$. Он соответствует символьному подканалу, замораживаемому при переходе к коду с r проверочными символами. □

Из доказательства теоремы видно, что статически замороженными являются все символьные подканалы поляризующего преобразования с номерами $i : \mathrm{wt}(i) < m - r$, где $r = \max_{0 \le j < k} \mathrm{wt}(j)$. Аналогичный подход был использован в [17, 93] для доказательства того, что расширенные коды БЧХ являются подкодами кодов Рида-Маллера.

Пример 3. *Рассмотрим* $(8, 4, 4)$ *код Рида-Соломона над* $\mathbb{F}_{2^3}$. *Его проверочная матрица имеет вид*

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & \alpha & 1+\alpha & \alpha^2 & 1+\alpha^2 & \alpha+\alpha^2 & 1+\alpha+\alpha^2 \\ 0 & 1 & \alpha^2 & 1+\alpha^2 & \alpha+\alpha^2 & 1+\alpha+\alpha^2 & \alpha & 1+\alpha \\ 0 & 1 & 1+\alpha & \alpha^2 & 1+\alpha^2 & \alpha+\alpha^2 & 1+\alpha+\alpha^2 & \alpha \end{pmatrix},$$

где α *— примитивный корень многочлена* $x^3 + x + 1$. *Применяя элементарные операции к строкам матрицы* $H(B_{2,3}F_2^{\otimes 3})^T$, *получим*

$$V = \begin{pmatrix} 0 & \alpha & 0 & 1+\alpha & 0 & \alpha+\alpha^2 & 1 & 0 \\ 0 & 1+\alpha & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \alpha+\alpha^2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Таким образом, символ u_0 *статически заморожен, а символы* u_2, u_4, u_6 *являются динамически замороженными.*

2.6.2. Метод последовательного исключения

Конструкция полярных кодов и метод последовательного исключения могут быть обобщены на случай произвольных конечных полей. Применение метода последовательного исключения в случае кодов над $\mathbb{F}_{2^\mu}, \mu > 1$, требует построения эффективной процедуры вычисления $\mathcal{W}_m^{(i)}(u_0^i | y_0^{n-1}), u_i \in \mathbb{F}_{2^\mu}$, где $u_i \in \mathbb{F}_{2^\mu}$ — входные символы поляризующего преобразования,

задаваемого $n \times n$ матрицей $A_m \in \mathbb{F}_2^{n \times n}$. Отметим, что выражение (1.11) в этом случае должно быть заменено на

$$\mathcal{W}_{\lambda}^{(2i)}(u_0^{2i}|y_0^{n-1}) = \sum_{u_{2i+1} \in \mathbb{F}_{2^\mu}} \mathcal{W}_{\lambda-1}^{(i)}(u_{0,e}^{2i+1} + u_{0,o}^{2i+1}|y_0^{\frac{n}{2}-1})\, \mathcal{W}_{\lambda-1}^{(i)}(u_{0,o}^{2i+1}|y_{\frac{n}{2}}^{n-1}). \tag{2.11}$$

Непосредственное его вычисление требует $2^{2\mu}$ операций. Отметим, что (2.11) представляет собой многомерную циклическую свертку. Поэтому более эффективная реализация процедуры расчета вероятностей $\mathcal{W}_{\lambda}^{(2i)}(u_0^{2i}|y_0^{n-1})$ со сложностью $O(\mu 2^\mu)$ может быть получена за счет использования быстрого преобразования Адамара. Кроме того, могут быть использованы методы ускорения вычислений, разработанные для случая недвоичных LDPC-кодов [50].

В случае передачи двоичного образа кода по симметричному каналу без памяти с двоичным входом получим, что

$$\mathcal{W}_{m}^{(i)}(u_0^i|y_0^{n-1}) = \prod_{j=0}^{\mu-1} \mathcal{W}_{m}^{(i)}(u_{0,j}^i|y_0^{n-1}), \tag{2.12}$$

где $u_{0,j}^i = (u_{0,j}, \ldots, u_{i,j})$, $u_s = \sum_{j=0}^{\mu-1} u_{s,j} a_j, u_{s,j} \in \mathbb{F}_2$, и $(a_0, \ldots, a_{\mu-1})$ — базис $\mathbb{F}_{2^\mu}$. Вероятности $\mathcal{W}_{m}^{(i)}(u_{0,j}^i|y_0^{n-1})$ могут быть вычислены с помощью выражений (1.11)–(1.12), после чего преобразованы в $\mathcal{W}_{m}^{(i)}(u_0^i|y_0^{n-1}), u_i \in \mathbb{F}_{2^\mu}$, что требует $O(2^\mu)$ операций.

Таким образом, декодирование расширенного кода Рида-Соломона может быть выполнено с помощью метода последовательного исключения (2.4). Вероятность ошибки при этом может быть оценена как

$$P \le \sum_{j \notin \mathcal{F}} \Pi_j,$$

где Π_j — вероятность ошибки в j-ом символьном подканале поляризующего преобразования при условии того, что значения $u_0, \ldots, u_{j-1}$ известны на стороне декодера. В общем случае при $\mu > 1$ задача нахождения Π_j пока не решена. Но для важного частного случая передачи двоичного образа кода по симметричному каналу без памяти с двоичным входом величины Π_j могут быть выражены через вероятности ошибки P_j, найденные для $\mu = 1$. Действительно, используя представление $u_i = \sum_{j=0}^{\mu-1} u_{i,j} a_j$, где $a_0, \ldots, a_{\mu-1}$ — некоторый базис $\mathbb{F}_{2^\mu}$, получим, что кодирование, передача и декодирование блоков $(u_{0,j}, \ldots, u_{n-1,j}), 0 \le j < \mu$, происходят независимо. Таким образом,

$$1 - \Pi_i = (1 - P_i)^\mu.$$

Величины P_i могут быть найдены с помощью метода эволюции плотностей [158] или гауссовской аппроксимации, описанной в разделе 3.1.1.1.

В работе [24] было показано, что пропускные способности подканалов $\mathcal{W}_{m}^{(i)}$ поляризующего преобразования $B_m F^{\otimes m}$ с входным алфавитом $\mathbb{F}_{2^\mu}$ в общем случае сходятся с увеличением m к величинам $0, 1, \ldots, \mu$. Для подканалов, пропускная способность которых близка к

$i \in \{1, \ldots, \mu - 1\}$, вероятность ошибки Π_j близка к $1 - 2^{i-\mu}$. В связи с этим могут существовать случаи, при которых многие величины $\Pi_j, j \notin \mathcal{F}$, достаточно велики, вследствие чего вероятность ошибки декодирования кода методом последовательного исключения неприемлемо высока. Однако для практически важного случая передачи двоичного образа кода по симметричному каналу без памяти справедлива оценка $P_i \leq Z_{m,i}$, где $Z_{m,i}$ задаются соотношениями (1.15)–(1.16). Можно заметить, что $Z_{m,i} = O(Z_{0,0}^{2^{\mathrm{wt}(i)}})$, где $\mathrm{wt}(i)$ — число ненулевых битов в двоичном представлении числа i и $Z_{0,0}$ — параметр Бхаттачарьи используемого канала передачи информации. Таким образом, вероятность ошибки декодирования двоичного образа некоторого кода над $\mathbb{F}_{2^\mu}$ методом последовательного исключения может быть оценена как $P \leq O(n\mu Z_{0,0}^{2^{w_0}})$, где $w_0 = \max_{i \notin \mathcal{F}} \mathrm{wt}(i)$.

Подробнее вопросы декодирования кодов Рида-Соломона с помощью предложенного подхода будут рассмотрены в разделе 4.5.

2.7. Выводы

В данной главе было введено обобщение классических конструкций полиномиальных и мономиальных кодов, называемое многочленными кодами. Оно позволяет представить любой линейный блоковый код в виде системы линейных ограничений динамического замораживания, налагаемых на вектор входных символов поляризующего преобразования. Предложенное обобщение позволяет воспользоваться методом последовательного исключения и его аналогами для декодирования произвольного линейного блокового кода. Вероятность ошибки декодирования при этом определяется расположением замороженных символов. В тех случаях, когда рассматриваемый код имеет достаточно низкоскоростной надкод Рида-Маллера, т.е. замороженными оказываются ненадежные подканалы соответствующего поляризующего преобразования, декодирование почти по максимуму правдоподобия может быть реализовано с помощью списочного или последовательного алгоритмов декодирования с небольшим размером списка. Были описаны множества замороженных символов для двоичных расширенных примитивных кодов БЧХ в узком смысле и кодов Рида-Соломона. Предложенное обобщение может также рассматриваться как расширение конструкции обобщенных каскадных кодов.

Было показано, что при декодировании расширенных кодов БЧХ длины 64 средняя сложность такого подхода оказывается меньше, чем при использовании алгоритма BEAST. Как будет показано в разделе 4.5, в случае кодов Рида-Соломона длины 31 предложенный подход позволяет получить лучшую корректирующую способность по сравнению с адаптив-

ным алгоритмом распространения доверия в сочетании с мягким алгебраическим декодированием. В разделе 4.3 будет показано, что в случае расширенного кода Голея средняя сложность алгоритма декодирования, основанного на предложенном представлении, сопоставима со сложностью алгоритма Варди, наилучшего известного алгоритма декодирования этого кода.

Глава 3

Полярные подкоды

В данной главе рассматривается задача построения кодов, которые допускали бы простое декодирование с помощью обобщений метода последовательного исключения и обладали бы при этом хорошей корректирующей способностью. Предлагаемая конструкция основана на введенном выше представлении линейных блоковых кодов в виде системы ограничений динамического замораживания. Кроме того, рассматривается задача упрощенного расчета надежностей подканалов поляризующего преобразования Арикана, возникающая при построении как классических полярных кодов, так и предлагаемого обобщения.

В разделе 3.1 представлены упрощенные методы расчета надежности подканалов поляризующего преобразования Арикана. Различные конструкции полярных подкодов описаны в разделе 3.2.

3.1. Расчет надежностей подканалов в поляризующем преобразовании Арикана

Как было отмечено в разделе 1.4.1, построение классического $(n = 2^m, k)$ полярного кода сводится к вычислению вероятностей ошибки P_i в битовых подканалах $\mathcal{W}_m^{(i)}$, задаваемых поляризующим преобразованием, и нахождению номеров $n - k$ подканалов с наибольшими значениями P_i. В случае полярных кодов Арикана это может быть сделано путем анализа плотностей распределения логарифмических отношений правдоподобия (1.13)–(1.13), возникающих в ходе работы декодера последовательного исключения. Однако метод эволюции плотностей [123, 158], реализующий данный подход, имеет сложность $O(n\mu^2 \log \mu)$, где μ — число уровней дискретизации каналов. Эксперименты показывают, что приемлемая точность достигается при $\mu \geq 256$, что делает этот метод слишком сложным для практического использования.

3.1.1. Аддитивный гауссовский канал

3.1.1.1. Метод Гауссовской аппроксимации

С практической точки зрения наиболее важен случай аддитивного гауссовского канала с дисперсией шума σ^2 и двоичной фазовой модуляции. Будем считать, что при модуляции символ 0 представляется как +1, а 1 — как −1. Пусть y_i — величины, наблюдаемые

на выходе канала с АБГШ при условии передачи по нему нулевого кодового слова. Тогда $L_0^{(0)}(y_i) \sim \mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$. В [46] было предложено аппроксимировать распределения промежуточных значений, возникающих в ходе работы алгоритма распространения доверия, использующего соотношения, аналогичные (1.13)–(1.14), гауссовскими. Это существенно упрощает анализ. Данный подход может быть обобщен на случай полярных кодов. Более конкретно, величины, задаваемые (1.13)–(1.14), могут рассматриваться как гауссовские случайные величины с $\mathbf{D}\left[L_\lambda^{(i)}\right] = 2\mathbf{M}\left[L_\lambda^{(i)}\right]$, где $\mathbf{M}[]$ и $\mathbf{D}[]$ — их матожидание и дисперсия, соответственно. Это позволяет вычислять только матожидание $\mathcal{L}_\lambda^{(i)} = \mathbf{M}\left[L_\lambda^{(i)}\right]$, значительно упрощая таким образом вычисления. В случае полярных кодов с ядром Арикана этот подход сводится к

$$\mathcal{L}_\lambda^{(2i)} = \Xi\left(\mathcal{L}_{\lambda-1}^{(i)}\right) = \phi^{-1}\left(1-\left(1-\phi\left(\mathcal{L}_{\lambda-1}^{(i)}\right)\right)^2\right) \tag{3.1}$$

$$\mathcal{L}_\lambda^{(2i+1)} = 2\mathcal{L}_{\lambda-1}^{(i)}, \tag{3.2}$$

$0 \le i < 2^{\lambda-1}$, где $\mathbf{M}\left[L_0^{(0)}\right] = \mathcal{L}_0^{(0)} = 2/\sigma^2$ и

$$\phi(x) = \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int_{-\infty}^{\infty} \tanh\frac{u}{2} e^{-\frac{(u-x)^2}{4x}} du, & x > 0 \\ 1, & x = 0. \end{cases}$$

Вероятность ошибки в каждом подканале может быть вычислена как

$$P_i \approx Q\left(\sqrt{\mathcal{L}_m^{(i)}/2}\right), 0 \le i \le 2^m - 1. \tag{3.3}$$

Видно, что вычисление $\mathcal{L}_m^{(i)}$ требует $\sum_{\lambda=0}^{m-1} 2^\lambda = 2^m - 1$ обращений к функции $\phi(x)$ и столько же операций умножения на 2.

Вычисление функции $\phi(x)$ вызывает на практике значительные затруднения. В [82] было предложено аппроксимировать эту функцию как

$$\phi(x) \approx \begin{cases} e^{0.0564x^2 - 0.4856x}, & 0 \le x < 0.867861 \\ e^{-0.4527x^{0.86} + 0.0218}, & 0.867861 \le x < 10 \\ \sqrt{\frac{\pi}{x}} e^{-x/4}\left(1 + \frac{1}{14x} - \frac{3}{2x}\right), & x \ge 10. \end{cases}$$

Однако это выражение также весьма сложно для практического использования.

Можно заметить (см. рис. 3.1), что функция $\Xi(x)$ является достаточно гладкой. С помощью метода наименьших квадратов была найдена аппроксимация

$$\Xi(x) \approx \begin{cases} 0.98611x - 2.31515 & x > 12 \\ x(9.0047 \cdot 10^{-3}x + 0.76943) - 0.95068, & 3.5 < x \le 12 \\ x(0.062883x + 0.36784) - 0.16267 & 1 < x \le 3.5 \\ x(0.22024x + 0.06448) & \text{иначе.} \end{cases} \tag{3.4}$$

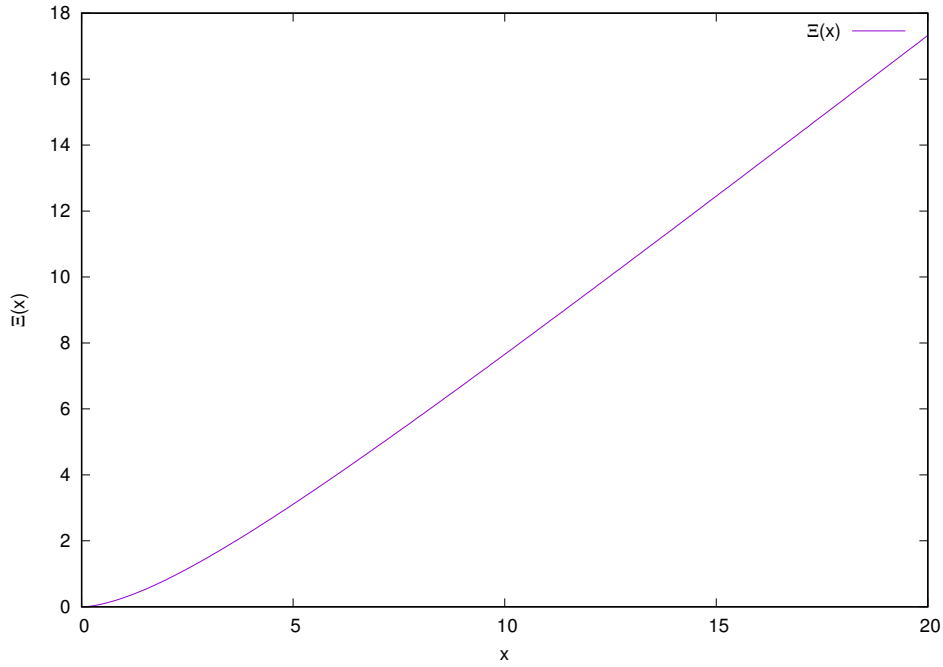


Рис. 3.1. Функция $\Xi(x)$

Таким образом, расчет надежности подканалов (т.е. величин P_i или $\mathcal{L}_m^{(i)}$) поляризующего преобразования Арикана требует не более $2n$ вещественных умножений и столько же сложений.

Для многих практических целей требуется построение укороченных и выколотых полярных кодов. Укорочение кода на символы $j_1, \ldots, j_s$ предполагает, что при кодировании символы $c_{j_i}, 1 \le i \le s$, всегда оказываются равными нулю, а потому опускаются при передаче. В приемнике опущенные символы восстанавливаются как абсолютно надежные, т.е. $\mathcal{L}_0^{(i)} = \mathbf{M}\left[L_0^{(0)}(y_{j_i})\right] = \infty$. Выкалывание также предполагает опускание при передаче символов с некоторыми номерами $h_1, \ldots, h_p$, однако в приемнике эти символы полагаются абсолютно ненадежными, т.е. $\mathcal{L}_0^{(i)} = \mathbf{M}\left[L_0^{(0)}(y_{h_i})\right] = 0$. В этом случае для расчета $\mathbf{M}\left[L_m^{(i)}\right] = \mathcal{L}_m^{(i)}$ можно использовать выражения

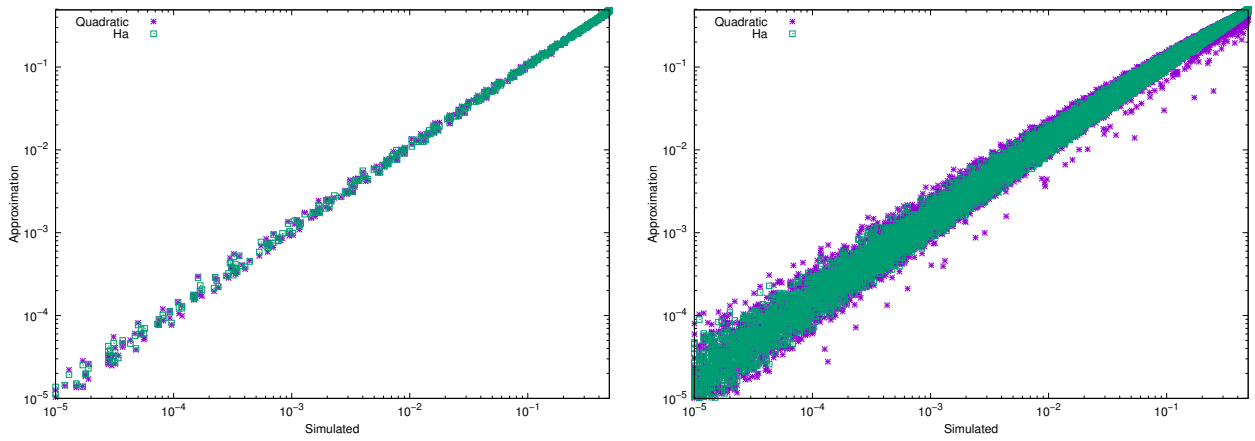
$$\mathcal{L}_\lambda^{(2i)} = \phi^{-1}\left(1 - \left(1 - \phi\left(\mathcal{L}_{\lambda-1}^{(i)}\right)\right)\left(1 - \phi\left(\mathcal{L}_{\lambda-1}^{(i+2^{m-\lambda})}\right)\right)\right) \tag{3.5}$$

$$\mathcal{L}_\lambda^{(2i+1)} = \mathcal{L}_{\lambda-1}^{(i)} + \mathcal{L}_{\lambda-1}^{(i+2^{m-\lambda})}, \tag{3.6}$$

которые являются непосредственным обобщением (3.1)–(3.2).

3.1.1.2. Численные результаты

На рисунке 3.2 представлены результаты статистического моделирования, иллюстрирующие точность гауссовской аппроксимации для подканалов поляризующего преобразования Арикана A_m. Рассматривался случай передачи данных по гауссовскому каналу с дисперсией шума $\sigma^2 = N_0/2 = 1$. Использовался алгоритм последовательного исключения, причем на всех



($а$) $m = 10$ ($б$) $m = 16$

Рис. 3.2. Точность гауссовской аппроксимации

его этапах при вычислении ЛОПП u_i использовались истинные значения $\widehat{u}_1^{i-1} = u_1^{i-1}$ с целью исключения размножения ошибок. Моделировалась передача 10^6 блоков данных. Каждая точка на графике соответствует одному из подканалов и представляет оценки вероятности ошибки в соответствующем подканале, найденные с помощью предложенной аппроксимации и экспериментально. Рассмотрены случаи реализации $\phi(x)$ с помощью аппроксимации Ха [82], а также предложенной кусочно-квадратичной аппроксимации $\Theta(x)$. Можно заметить, что метод гауссовской аппроксимации дает достаточно точные результаты. Несоответствие теоретических и экспериментальных результатов в области малых вероятностей ошибки связано с недостаточным объемом выборки.

В таблице 3.1 представлены значения среднего квадрата ошибки оценивания натурального логарифма вероятности ошибки в битовых подканалах (см. (3.3)) при использовании предложенной аппроскимации. Учитывались только подканалы с вероятностью ошибки (найденной экспериментально) не менее 10^{-4}. Видно, что использование кусочно-квадратичной аппроксимации при $m \leq 13$ дает примерно ту же точность, что и использование аппроксимации Ха.

Таблица 3.1. Погрешность гауссовской аппроксимации

m	Аппроксимация Ха	Кусочно-квадратичная аппроксимация
1	0.00012	7.19e-005
2	0.00036	0.00031
3	0.00057	0.00057
4	0.00072	0.00087
5	0.0022	0.0016
6	0.0027	0.002
7	0.0037	0.0033
8	0.0043	0.0050
9	0.0074	0.0084
10	0.0081	0.010
13	0.017	0.023
16	0.034	0.051

3.1.2. Релеевский канал

Рассмотрим передачу символов двоичной фазовой модуляции по каналу с независимыми релеевскими замираниями. В этом случае принятый сигнал может быть представлен[1] как

$$y_j = h_j(-1)^{x_j} + \eta_j, \tag{3.7}$$

где h_j распределены по закону Релея и $\eta_i \sim \mathcal{N}(0, \sigma^2)$. Будем считать случайные величины h_j и η_j независимыми. Распределение Релея является частным случае χ распределения с $\nu = 2$ степенями свободы, которое имеет плотность распределения

$$\psi(x, \nu) = \frac{2x^{\nu-1}e^{-x^2}}{\Gamma(\nu/2)}. \tag{3.8}$$

Заметим, что в общем случае ν может быть нецелым.

Логарифмические отношения правдоподобия принятых символов равны $L_0^{(0)}(y_j) = z_j = \frac{2h_j y_j}{\sigma^2}$. Если по каналу передается последовательность $x_j = 0$, получим, что условная плотность распределения логарифмических отношений правдоподобия имеет вид [160]

$$f_{LLR}(z|x_j = 0) = \frac{\sigma^2}{2\sqrt{1+2\sigma^2}} e^{\frac{z-|z|\sqrt{1+2\sigma^2}}{2}} \tag{3.9}$$

[1] В литературе при описании этой модели зачастую предполагают, что h_j и η_j являются комплексными гауссовскими случайными величинами. Такое описание может быть сведено к представленному здесь путем ортогонального преобразования квадратурной и синфазных компонент y_j.

Таблица 3.2. Параметры $(\Lambda_m^{(i)}, \sigma_m^{(i)})$ битовых подканалов 8-мерного поляризующего преобразования

m \ i	0	1	2	3	4	5	6	7
0	$(1, 0.5)$							
1	$(1.03, 0.8)$	$(2, 0.5)$						
2	$(1.07, 1.4)$	$(2.06, 0.8)$	$(1.95, 0.63)$	$(4, 0.5)$				
3	$(1.11, 3.05)$	$(2.14, 1.4)$	$(1.77, 1.01)$	$(4.12, 0.8)$	$(1.81, 0.8)$	$(3.9, 0.63)$	$(4.04, 0.57)$	$(8, 0.5)$

3.1.2.1. Двумерный метод

Распределение (3.9) существенно отличается от нормального. Применение метода гауссовской аппроксимации для расчета вероятностей ошибки в битовых подканалах поляризующего преобразования Арикана приводит к весьма значительной погрешности. В данном разделе представлен альтернативный метод решения данной задачи.

Заметим, что выражение (1.14) может рассматриваться как реализация техники разнесенного приема [142]. Если $r_j = h_j s + \eta_j, 0 \le j < \Lambda$, являются откликами канала на один и тот же переданный символ $s \in \{-1, 1\}$, оценка последнего может быть найдена как

$$\widehat{s} = \sum_{j=0}^{\Lambda-1} h_j r_j = \underbrace{\left(\sum_{j=0}^{\Lambda-1} h_j^2\right)}_{\tilde{h}^2} s + \underbrace{\sum_{j=0}^{\Lambda-1} h_j \eta_j}_{\eta}.$$

Если h_j распределены по закону χ с 2 степенями свободы, то $\tilde{h}^2$ распределено по закону χ^2 с 2Λ степенями свободы, в то время как η распределено по нормальному закону с дисперсией $\tilde{h}^2\sigma^2$. Можно показать, что вероятность ошибки при использовании данного метода равна [142]

$$\begin{aligned} P(\Lambda, \sigma^2) &= \left(\frac{1-\mu}{2}\right)^{\Lambda} \sum_{k=0}^{\Lambda-1} C_{\Lambda-1+k}^{k} \left(\frac{1+\mu}{2}\right)^{k} \\ &= 1 - \left(\frac{1-\mu^2}{4}\right)^{\Lambda} {}_2F_1\left[\begin{matrix} 1 & 2\Lambda \\ \Lambda+1 \end{matrix}; \frac{1+\mu}{2}\right] C_{2\Lambda-1}^{\Lambda} \end{aligned} \tag{3.10}$$

где $\mu = \frac{1}{\sqrt{1+2\sigma^2}}$, и ${}_2F_1\left[\begin{smallmatrix} a & b \\ c \end{smallmatrix}; z\right]$ — обобщенная гипергеометрическая функция. Эффективные методы ее расчета приведены в [33], и их реализация доступна во многих численных библиотеках. Для достаточно малых σ это выражение может быть аппроксимировано

$$P(\Lambda, \sigma^2) \approx \left(\frac{\sigma^2}{2}\right)^{\Lambda} C_{2\Lambda-1}^{\Lambda}.$$

Ввиду доступности относительно простого выражения для расчета вероятности ошибки (3.10), предлагается моделировать подканалы поляризующего преобразования как в (3.7),

полагая, что h_j распределено по закону χ с $2\Lambda_m^{(i)}$ степенями свободы, и $\eta_i \sim \mathcal{N}(0, (\sigma_m^{(i)})^2)$, где m и i —номера слоя и фазы, соответственно, причем $\Lambda_0^{(0)} = 1, \sigma_0^{(0)} = \sigma$. Таким образом, каждый подканал может быть охарактеризован парой $(\Lambda_m^{(i)}, \sigma_m^{(i)})$. Из (1.14) можно заметить, что

$$\Lambda_{m+1}^{(2i+1)} = 2\Lambda_m^{(i)} \tag{3.11}$$

$$\sigma_{m+1}^{(2i+1)} = \sigma_m^{(i)}. \tag{3.12}$$

Таким образом, комбинируя два подканала поляризующего преобразования с параметрами (Λ, σ) в соответствии с (1.12) или (1.14), можно получить аналогичный подканал с некоторыми другими значениями параметров (Λ', σ'). Будем считать, что это же верно и для случая использования преобразования (1.11) или (1.13). Для того, чтобы получить формулы для расчета параметров подканалов в данном случае, введем функцию

$$C_t(\Lambda_m^{(i)}, \sigma_m^{(i)}) = E\left[\tanh^t\left(\frac{L_m^{(i)}}{2}\right)\right] = E\left[\tanh^t\left(\frac{hy}{(\sigma_m^{(i)})^2}\right)\right]$$
$$= \int_0^\infty \int_{-\infty}^\infty \tanh^t\left(\frac{hy}{(\sigma_m^{(i)})^2}\right)\psi(h, 2\Lambda_m^{(i)})\phi(y-h, \sigma_m^{(i)})dydh,$$

где $\phi(y, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}}$, $t \in \{1, 2\}$. Тогда из (1.13) получим

$$C_t(\Lambda_{m+1}^{(2i)}, \sigma_{m+1}^{(2i)}) = C_t(\Lambda_m^{(i)}, \sigma_m^{(i)})^2, t = 1, 2. \tag{3.13}$$

Для нахождения параметров подканалов на слое $m+1$ и фазе $2i$ эта система уравнений может быть решена численно. Заметим, что в общем случае это приводит к нецелочисленным значениям $\Lambda_{m+1}^{(2i)}$.

Пример 4. *В таблице 3.2 приведены параметры битовых подканалов для случая поляризующего преобразования Арикана* 8×8 *и* $\sigma = 0.5$.

3.1.2.2. Упрощенный одномерный метод

Численное решение (3.13) является вычислительно сложной задачей, т.к. включает многократное вычисление двумерных интегралов. В связи с этим представляет интерес упрощенный метод.

Рис. 3.3 иллюстрирует зависимость Λ' от Λ, где $C_t(\Lambda', \sigma') = C_t(\Lambda, \sigma)^2, t = 1, 2$. Можно заметить, что при достаточно малых σ выполняется $\Lambda' \approx \Lambda$. В связи с этим предлагается установить $\Lambda_{m+1}^{(2i)} = \Lambda_m^{(i)}$, и ограничиться вычислением только $\sigma_{m+1}^{(2i)}$. Пусть $p_{m,i} = P(\Lambda_m^{(i)}, \sigma_m^{(i)})$ — вероятность ошибки в подканале $W_m^{(i)}(y|x)$. Несложно убедиться [70, 106] в том, что $p_{m+1,2i} =$

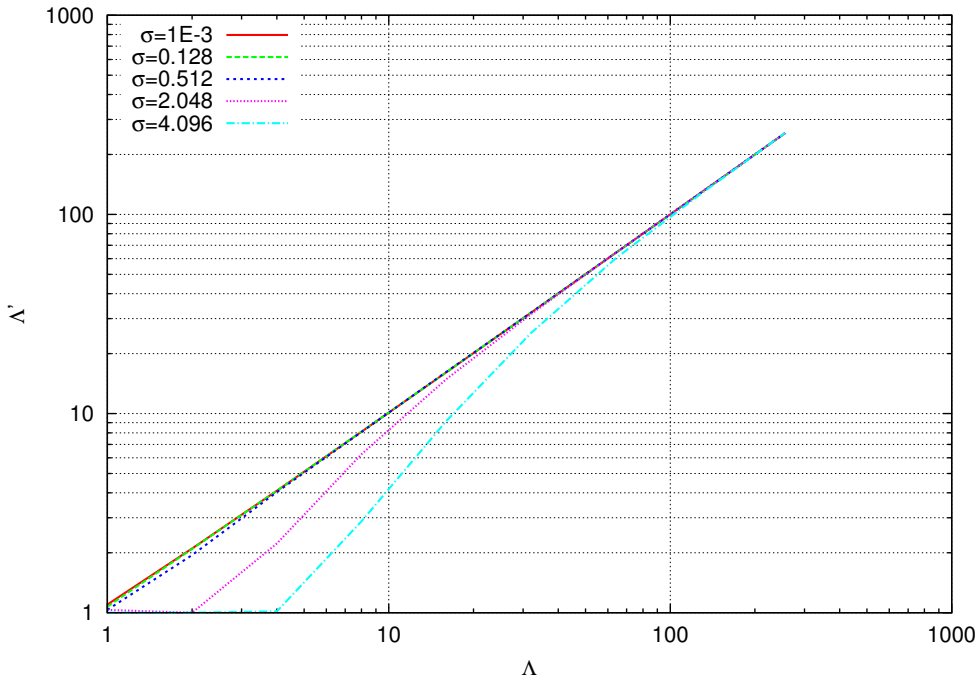


Рис. 3.3. $\Lambda' : C_t(\Lambda', \sigma') = C_t(\Lambda, \sigma)^2$

$2p_{m,i} - 2p_{m,i}^2$. Вычислив $p_{m+1,2i}$, можно найти соответствующее значение $\sigma_{m+1}^{(2i)}$ путем численного решения уравнения (3.10). Таким образом, получим следующее рекурсивное выражение для $\sigma_m^{(i)}$:

$$P(\Lambda_{m+1}^{(2i)}, \sigma_{m+1}^{(2i)}) = 2P(\Lambda_m^{(i)}, \sigma_m^{(2i)})(1 - P(\Lambda_m^{(i)}, \sigma_m^{(i)})). \tag{3.14}$$

В этом случае выполняется также

$$\Lambda_m^{(i)} = 2^{\mathrm{wt}(i)}, \tag{3.15}$$

где $\mathrm{wt}(i)$ — число ненулевых битов в двоичном представлении числа i. Параметры подканалов с нечетными номерами могут быть найдены с помощью выражений (3.11)–(3.12).

Пример 5. *В таблице 3.3 представлены приближенные значения $\sigma_m^{(i)}$ в случае 8-мерного поляризующего преобразования Арикана и $\sigma = 0.5$. Видно, что эти значения достаточно близки к тем, которые приведены в таблице 3.2.*

Приведенные результаты свидетельствуют о том, что логарифмические отношения правдоподобия $L_1^{(0)}$, которые получаются из принятых зашумленных символов путем однократного применения (1.13), могут рассматриваться как случайные величины с плотностью распределения, задаваемой (3.9), где $\sigma = 0.79$. На рис. 3.4 представлена соответствующая диаграмма квантиль-квантиль. Можно заметить, что распределение экспериментальных данных достаточно близко к предложенной модели.

Таблица 3.3. Приближенные значения $\sigma_m^{(i)}$ для битовых подканалов $A = B_{2,3}F_2^{\otimes 3}$

m \ i	0	1	2	3	4	5	6	7
0	0.5							
1	0.79	0.5						
2	1.43	0.79	0.64	0.5				
3	3.51	1.42	1.12	0.79	0.86	0.64	0.57	0.5

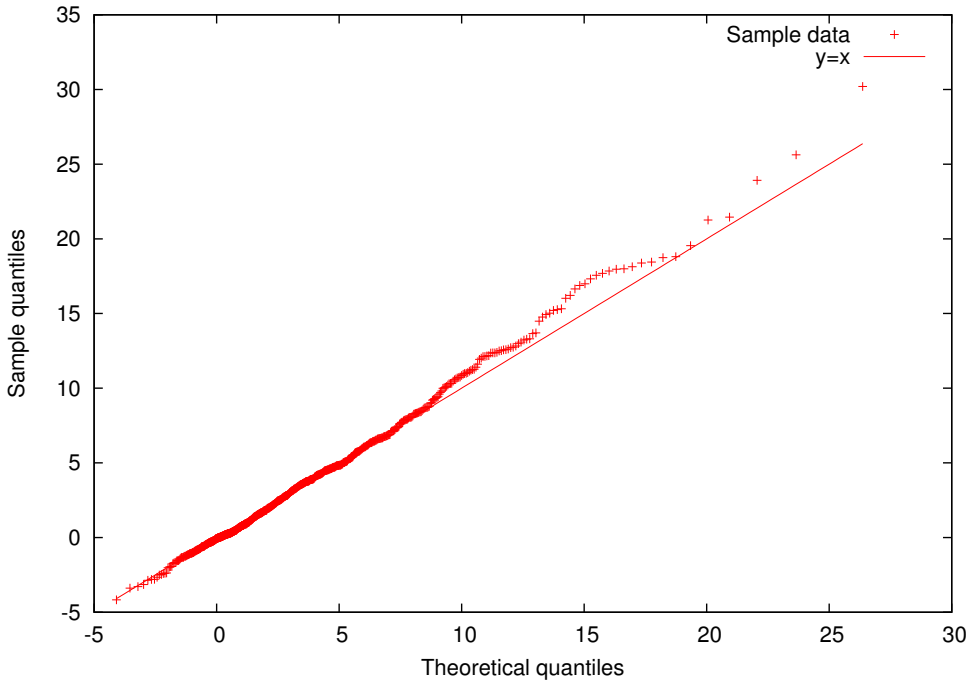


Рис. 3.4. График квантиль-квантиль для $L_1^{(0)}$

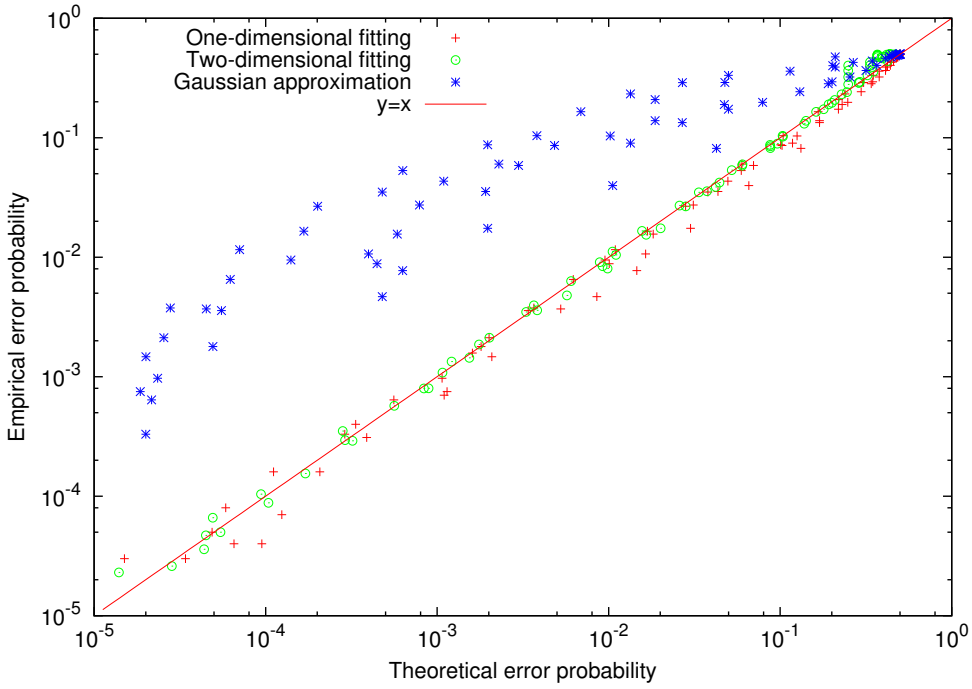


Рис. 3.5. Вероятности ошибки в битовых подканалах для $m = 7, \sigma = 0.5$

3.1.2.3. Численные результаты

На рис. 3.5 представлены найденные теоретически и экспериментально вероятности ошибки в битовых каналах поляризующего преобразования Арикана размерности A_7 для случая $\sigma = 0.5$. Рассмотрены как двумерный метод подбора параметров распределения, задаваемый (3.11)–(3.13), так и одномерный метод, описываемый (3.12), (3.14), (3.15). Кроме того, приведены результаты, полученные с помощью альтернативного метода гауссовской аппроксимации, основанного на работе [106]. Можно заметить, что двумерный метод обеспечивает достаточно точные результаты во всех случаях за исключением самых ненадежных подканалов, где численное решение (3.13) представляет большую сложность. Одномерный метод дает несколько завышенные оценки вероятности ошибки в надежных подканалах. Можно также заметить, что гауссовская аппроксимация непригодна для построения полярных кодов в канале с релеевскими замираниями.

На рис. 3.6 представлена корректирующая способность $(1024, 512, d)$ полярных подкодов (см. раздел 3.2) в случае декодирования методом ПИ и последовательным алгоритмом [119] с различными значениями размера списка l. Результаты представлены для кодов, построенных для релеевского канала с $E_b/N_0 = 5dB$ и гауссовского канала с $E_b/N_0 = 2dB$. Эти коды обозначены номерами R и F, соответственно. При построении этих кодов использовался одномерный метод расчета вероятности ошибки в битовых подканалах. Код с минимальным расстоянием $d = 16$ является классическим полярным кодом Арикана, в то время как коды с

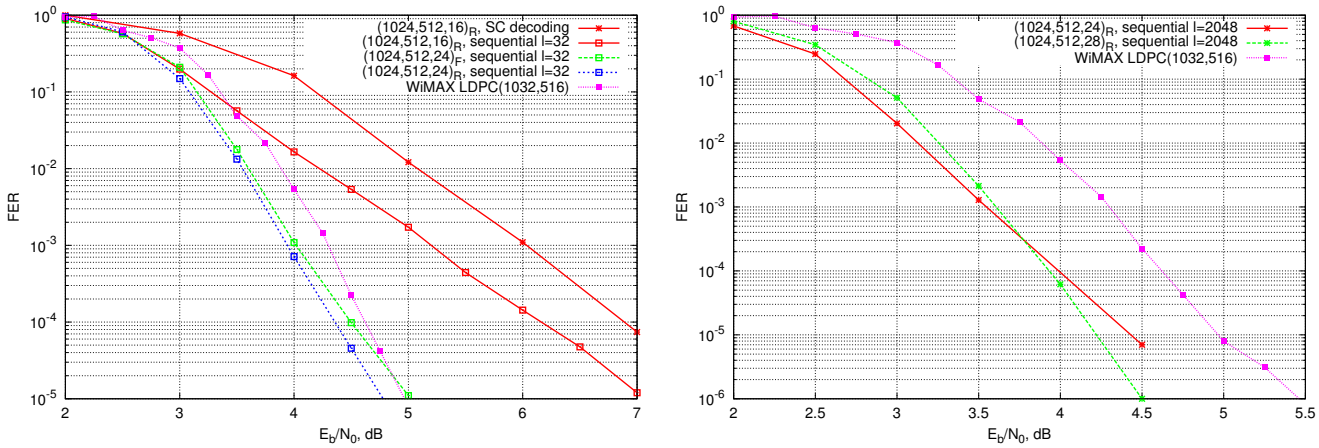


Рис. 3.6. Корректирующая способность полярных (под)кодов в Релеевском канале

минимальными расстояниями $d = 24$ и $d = 28$ являются полярными подкодами кодов РБЧХ (см. раздел 3.2.1.1). Для сравнения приведены результаты для $(1032, 516)$ LDPC-кода. В последнем случае декодирование выполнялось с помощью алгоритма распространения доверия с 200 итераций. Можно заметить, что корректирующая способность классических полярных кодов весьма плоха даже при использовании последовательного декодирования. Полярные подкоды дают существенно лучшие результаты. Заметим, что коды, оптимизированные для релеевского канала, обеспечивают лучшую корректирующую способность по сравнению с кодами, построенными для гауссовского канала. Можно заметить, что при достаточно большом l рассмотренные коды обеспечивают энергетический выигрыш до 0.8 дБ по сравнению с LDPC кодом.

подканалы поляризующего преобразования как в (3.7), полагая,

3.2. Полярные подкоды

Можно показать, что минимальное расстояние полярных кодов с ядром Арикана равно $O(\sqrt{n})$ [87]. В результате их корректирующая способность при декодировании по максимуму правдоподобия оказывается неудовлетворительной. С другой стороны, наличие вычислительно простых алгоритмов декодирования делает эти коды крайне привлекательными для практического использования. Вместе с тем, строгий анализ корректирующей способности списочного алгоритма Тала-Варди и его аналогов остается открытой проблемой. Вероятность ошибки этого алгоритма может быть выражена как

$$P(L) = P_{ML} + P\left\{E(L)|\mathbb{C}\right\},$$

где P_{ML} — вероятность ошибки декодирования рассматриваемого кода по максимуму правдоподобия, $\mathbb{C}$ — событие, соответствующее принятию декодером по максимуму правдоподобия правильного решения, и $E(L)$ — вероятность того, что метрика правильного пути оказывается меньше метрик L неправильных путей на одной из промежуточных фаз, что приводит к ошибке списочного/последовательного декодирования. Величина P_{ML} может быть оценена с использованием весового спектра кода. На высоких отношениях сигнал/шум P_{ML} в основном зависит от минимального расстояния d кода и его ошибочного коэффициента, т.е. числа w_d кодовых слов веса d. Насколько известно автору, до настоящего времени не разработаны методы оценки вероятности $P\{E(L)|\mathbb{C}\}$. Однако эксперименты показывают, что эта величина убывает с уменьшением P_{SC}, вероятности ошибки декодирования кода методом ПИ.

В связи с этим представляет интерес построение кодов, которые одновременно обладали бы низкой вероятностью ошибки декодирования методом последовательного исключения P_{SC}, достаточно большим минимальным расстоянием d и сравнительно небольшим числом w_d кодовых слов веса d.

Определение 7 ([178, 19]). *Рассмотрим q-ичный симметричный канал $\mathcal{W}(y|c)$ и $(n = l^m, k', d)$ код $\mathcal{C}'$ над $\mathbb{F}_q$, называемый протокодом. Пусть $\mathcal{F}'$ — множество номеров замороженных символов $\mathcal{C}'$ относительно ядра F_l. $(n, k, \geq d)$ полярным подкодом в узком смысле $\mathcal{C}$ кода $\mathcal{C}'$ называется множество векторов $c_0^{n-1} = u_0^{n-1} B_{l,m} F_l^{\otimes m}$, где u_0^{n-1} одновременно удовлетворяет ограничениям (2.3) кода $\mathcal{C}'$, а также дополнительным ограничениям $u_s = 0$ для $k' - k$ номеров $s \notin \mathcal{F}'$ с наибольшими вероятностями ошибки P_s.*

Кодирование в полярном подкоде в узком смысле может быть выполнено как

$$c = xW B_{l,m} F_l^{\otimes m}, \tag{3.16}$$

где x — информационный вектор, W — матрица, удовлетворяющая $WV^T = 0$, и V — матрица ограничений динамического замораживания.

Ниже будет предложена несколько более общая конструкция кодов. Эти коды можно рассматривать как подкоды кодов, полученных чередованием нескольких полярных кодов. Для удобства будем их также называть полярными подкодами.

Определение 8. *Пусть дано ядро F_l и натуральные числа $m_0, \ldots, m_{z-1}$. Полярным подкодом в широком смысле называется множество векторов*

$$c = xW \operatorname{diag}(B_{l,m_0} F_l^{\otimes m_0}, \ldots, B_{l,m_{z-1}} F_l^{\otimes m_{z-1}}), x \in \mathbb{F}_q^k,$$

где матрица W имеет нулевые столбцы в позициях, соответствующих r подканалам $\mathcal{W}_{m_t}^{(j)}, 0 \leq t < z$, с наибольшей вероятностью ошибки. Здесь z и r — параметры конструкции.

Полярные коды с CRC [157] и внешними кодами LDPC [60] могут рассматриваться как частные случаи этой конструкции при $z = 1$.

Следует понимать, что при декодировании методом ПИ полярные подкоды как в узком, так и широком смысле не имеют никаких преимуществ перед классическими полярными кодами, построенными для соответствующего канала. Это связано с тем, что вероятность ошибки декодирования методом ПИ (1.10) однозначно определяется множеством номеров замороженных символов, и классическая конструкция полярных кодов явным образом минимизирует эту величину. Вместе с тем, как будет показано ниже, полярные подкоды демонстрируют существенный выигрыш при использовании списочного и последовательного алгоритмов декодирования.

3.2.1. Полярные подкоды с одним поляризующим преобразованием

3.2.1.1. Полярные подкоды кодов РБЧХ

Рассмотрим построение полярных подкодов в узком смысле с заданным минимальным расстоянием d.

1. Ядро Арикана. Из вышеприведенных соображений следует, что в качестве кода $\mathcal{C}'$ следует выбирать такой код, что его множество $\mathcal{F}'$ включало бы как можно больше номеров подканалов с высокой вероятностью ошибки P_{j_i}.

Из теоремы 2 видно, что номера замороженных символов кода РБЧХ имеют достаточно малый вес. Из границ (1.15)–(1.16) следует, что параметр Бхаттачарьи i-го подканала удовлетворяет $Z_{n,i} = O(Z_{1,0}^{2^{\mathrm{wt}(i)}})$. Таким образом, использование кодов РБЧХ в качестве протокодов в предложенной конструкции позволяет избежать замораживания большого числа символов, соответствующих подканалам с малым $Z_{n,i}$, и обеспечивает достаточно низкое значение P_{SC}.

Заметим, что увеличение минимального расстояния протокода приводит к замораживанию большего числа символов, соответствующих подканалам с малым $Z_{n,i}$. При фиксированной размерности полярного подкода это требует размораживания символов, соответствующих подканалам с большим $Z_{n,i}$, что приводит к увеличению P_{SC}. Это может быть компенсировано увеличением размера списка в алгоритме Тала-Варди и его аналогах. Ввиду отсутствия аналитических способов расчета его вероятности ошибки, оптимальное значение минимального расстояния d протокода для случая декодирования с заданным размером списка L приходится подбирать экспериментально.

Пример 6. *Построим полярный* $(16, 6, 6)$ *подкод* $(16, 7, 6)$ *кода РБЧХ, рассмотренного*

в примере 2. Будем оптимизировать подкод для двоичного стирающего канала с вероятностью стирания $Z_{1,0} = 0.5$. *Вектор параметров Бхаттачарьи (т.е. вероятностей стирания) поляризующего преобразования имеет вид* $Z_{16} = (\underline{0.999}, \underline{0.992}, \underline{0.985}, 0.77, \underline{0.96}, \underline{0.65}, 0.53, 0.1, \underline{0.9}, \underline{0.47}, \underline{0.35}, 3.7 \cdot 10^{-2}, \underline{0.23}, 1.5 \cdot 10^{-2}, 7.8 \cdot 10^{-3}, 1.5 \cdot 10^{-5})$. *Здесь величины, соответствующие замороженным символам кода РБЧХ подчеркнуты. Можно заметить, что* u_3 *имеет наивысшую вероятность стирания* 0.77 *среди незамороженных символов. В связи с этим предлагается ввести дополнительное ограничение* $u_3 = 0$. *Это эквивалентно удалению первой строки из матриц* G_1 *и* G_3 *в примере 1.*

Пример 7. *Рассмотрим построение* $(1024, 512)$ *полярного подкода. Существует* $(1024, 513, 116)$ *код РБЧХ, который не может быть продекодирован с приемлемой сложностью и корректирующей способностью с использованием списочного алгоритма Тала-Варди. Классический полярный код, построенный для канала с АБГШ с двоичной фазовой модуляцией и* $E_b/N_0 = 2$ *дБ имеет минимальное расстояние 16. Выбирая* $(1024, 893, 28)$ *код РБЧХ в качестве протокода* $\mathcal{C}' : RM(5, 10) \subset \mathcal{C}' \subset RM(8, 10)$, *и вводя* 381 *дополнительных ограничения замораживания, получим* $(1024, 512, \geq 28)$ *полярный подкод. Описание этого кода включает лишь* $f = 20$ *нетривиальных уравнений* (2.3) *с* $T = 111$ *членами, т.е. сложность вычисления значений ДЗС оказывается пренебрежимо малой по сравнению со сложностью умножения на матрицу* A_m.

Заметим, что вероятность ошибки P декодирования методом ПИ (n, k, d) любого полярного подкода не может быть меньше, чем вероятность ошибки $\overline{P}$ декодирования тем же методом классического полярного (n, k) кода с тем же ядром, построенного для того же канала. Однако в случае использования списочного метода декодирования с достаточно большим размером списка L корректирующая способность полярного подкода может быть существенно лучше. Было замечено, что размер списка L, необходимый для получения такого выигрыша, растет с P. Таким образом, необходимо оценить значение $\overline{P}/P$, требуемое для достижения заданного минимального расстояния d. Однако проще найти скорость полярного подкода с заданным минимальным расстоянием, имеющего ту же вероятность ошибки декодирования методом ПИ, что и классический полярный код той же длины.

Пусть $\overline{\mathcal{C}}$ — полярный код с ядром F_2 и скоростью $\overline{\rho}(z)$, причем все символы с $Z_{m,i} < z$ не заморожены. Рассмотрим (n, k, d) полярный подкод $\mathcal{C}$ со скоростью $\rho(z, d) = k/n$, полученный из $(n = 2^m, k' = \beta(m, d)n, d)$ кода РБЧХ $\mathcal{C}'$ путем замораживания всех символов u_i с $Z_{m,i} \geq z$. Множество номеров незамороженных символов кода $\mathcal{C}$ может быть представлено как $\Delta(d, z) = \cup_{r=0}^{m-1} \left(\Delta(r, m, z) \setminus \mathcal{F}'_{d,r}\right)$, где $\Delta(r, m, z) = \{i | 0 \leq i < 2^m, \mathrm{wt}(i) = r, Z_{2^m,i} < z\}$, и $\mathcal{F}'_{d,r}$ —

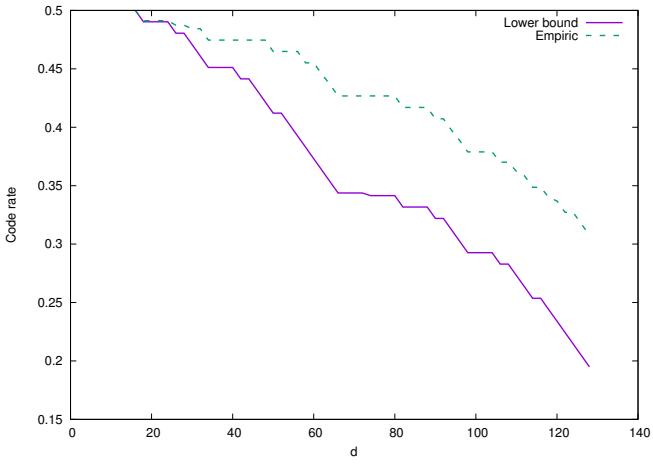


Рис. 3.7. Скорость полярных подкодов длины 1024

множество номеров замороженных символов u_e кода $\mathcal{C}'$ таких, что $\mathrm{wt}(e) = r$.

Достаточно сложно найти $|\Delta(r, m, z)|$ аналитически. Однако эта величина может быть найдена численно за полиномиальное время для любого линейного кода и симметричного канала с двоичным входом, а также величин r, m [158]. Ее можно также аппроксимировать асимптотическим выражением для доли общих незамороженных символов кода Рида-Маллера со скоростью $R(m-r, m) = 2^{-m} \sum_{j=0}^{m-r} \binom{m}{j}$ и полярного кода длины $n = 2^m$ со скоростью ρ. В [144] показано, что эта доля при $m \to \infty$ стремится к $\phi(\rho, r, m) = C \min\left(\frac{\rho}{C}, R(m-r, m)\right)$, где C — пропускная способность рассматриваемого канала. Таким образом, $2^{-m}|\Delta(r, m, z)| \approx \phi(\overline{\rho}(z), r, m) - \phi(\overline{\rho}(z), r+1, m)$.

Отсюда вытекает, что

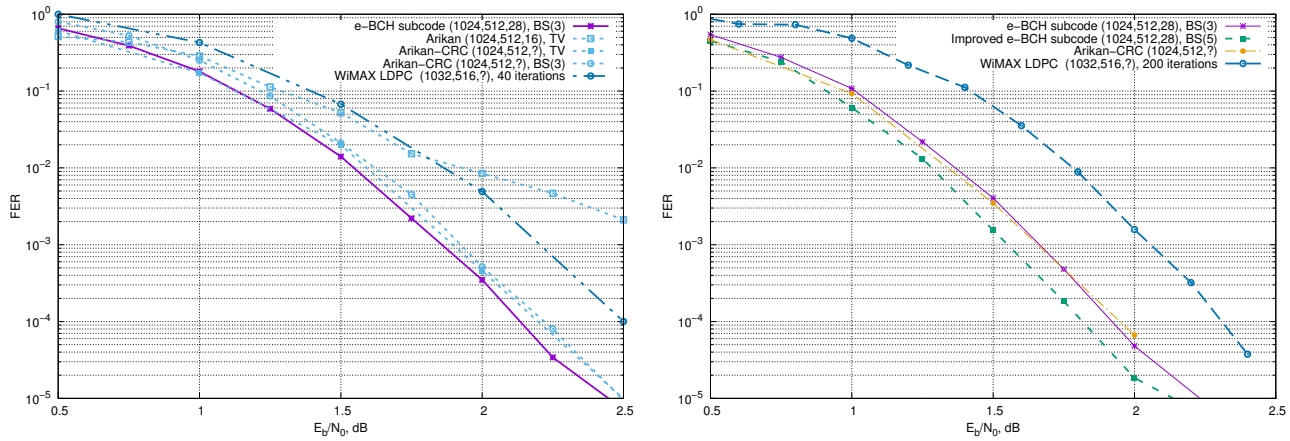
$$|\Delta(r, m, z) \setminus \mathcal{F}'_{d,r}| \geq \max(0, |\Delta(r, m, z)| - |\mathcal{F}'_{d,r}|), \tag{3.17}$$

и

$$\rho(z, d) \geq 2^{-m} \sum_{r=0}^{m} \max(0, |\Delta(r, m, z)| - |\mathcal{F}'_{d,r}|) \approx$$
$$\approx \sum_{r=0}^{m} \max\left(0, \phi(\overline{\rho}(z), r, m) - \phi(\overline{\rho}(z), r+1, m) - N_r 2^{-m}\right),$$

где N_r задается теоремой 2.

На рис. 3.7 приведена эта граница вместе с истинной скоростью (n, k, d) полярных подкодов кодов РБЧХ длины $n = 1024$. Размерность k этих подкодов была выбрана таким образом, чтобы они достигали примерно той же вероятности ошибки декодирования методом ПИ на $E_s/N_0 = -1$ дБ, как и классические полярный код Арикана $(1024, 512, 16)$, построенный для того же значения E_s/N_0. Видно, что эта граница весьма неточна. Это обусловлено как использованием (3.17), которое предполагает, что все замороженные символы, задаваемые



(*а*) $L = 32$ (*б*) $L = 256$

Рис. 3.8. Корректирующая способность полярных подкодов с ядром Арикана кодов РБЧХ длины 1024

протокодом РБЧХ, соответствуют подканалам с наименьшими возможными параметрами Бхаттачарьи, так и использованием асимптотической формулы для $|\Delta(r, m, z)|$.

Можно заметить, что ухудшение скорости полярного подкода по сравнению с классическим полярным кодом остается незначительным при d, не превосходящем удвоенного минимального расстояния классического полярного кода с той же вероятностью ошибки декодирования методом ПИ.

Кодирование предлагаемых кодов может быть выполнено с помощью (3.16) со сложностью $C_W + \frac{1}{2} n \log n$, где $C_W = T - f$ — сложность умножения на матрицу W, T — число членов в правой части в нетривиальных уравнениях (2.3), и $f \leq \sum_t N_t$ — число таких уравнений. Систематическое кодирование может быть выполнено с помощью метода, предложенного в [66] со сложностью $C_W + n \log n$.

Из теоремы 2 видно, что множество номеров замороженных символов протокода РБЧХ включает числа с достаточно малым весом. Можно также заметить, что ограничение динамического замораживания для символа u_i не может включать символы $u_j : \mathrm{wt}(j) > \mathrm{wt}(i)$. С другой стороны, большая часть ограничений $u_s = 0$, налагаемых на символы, соответствующие подканалам с высокой вероятностью ошибки P_s, относится к числам s малого веса. В связи с этим f оказывается существенно меньше, чем значение $\sum_t N_t$, вытекающее из теоремы 2 для протокода, т.е. матрица W является разреженной. Это было проиллюстрировано в примере 7. Таким образом, сложность кодирования предлагаемых полярных подкодов не превосходит таковую для полярных кодов с f-битным CRC.

На рис. 3.8 представлены результаты статистического моделирования для кодов длины

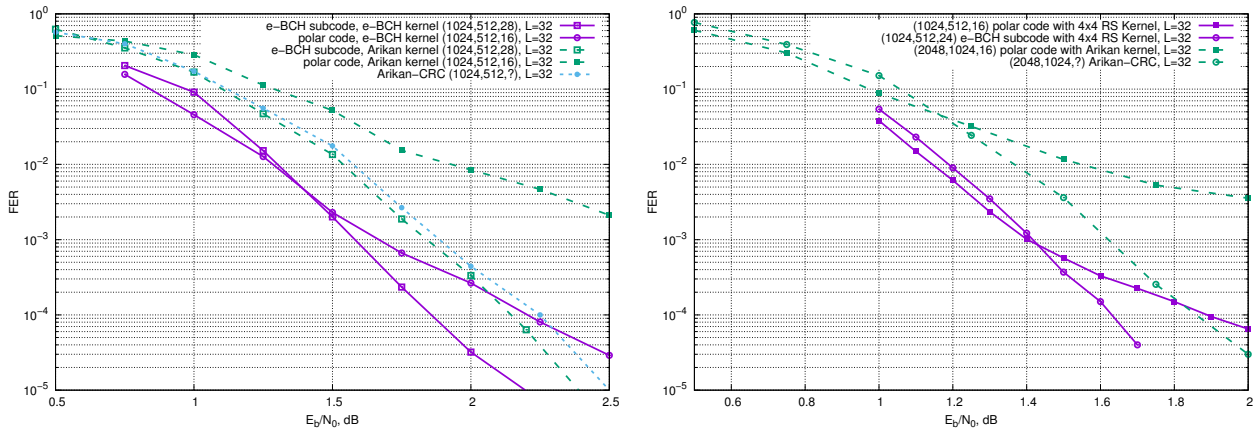
≈ 1024 для случая передачи по каналу с АБГШ и двоичной фазовой модуляцией. Для сравнения приведены также результаты для полярных кодов (Arikan) и полярных кодов с CRC-16 (Arikan-CRC) [157], а также LDPC кода из стандарта WiMAX. Декодирование полярных (под)кодов выполнялось с помощью блочного последовательного алгоритма декодирования (BS(s)) (см. раздел 4.1.3), где 2^s — длина внешних кодов в представлении рассматриваемого кода как ОККПС.

Можно заметить, что полярный подкод обеспечивает значительный энергетический выигрыш по сравнению с классическим полярным кодом, полярным кодом с CRC и LDPC-кодом.

2. Ядро РБЧХ. Пусть $C(q,s,m,J)$ — код над $\mathbb{F}_q$ длины q^{sm}, состоящий из векторов значений многочленов от m переменных $\frac{1}{(q^s-1)^m}A(x_0,\ldots,x_{m-1})$ в различных точках $(x_0,\ldots,x_{m-1}) \in \mathbb{F}_{q^s}^m$, причем коэффициенты $A_{j_0,\ldots,j_{m-1}}$ этих многочленов равны нулю при $(j_0,\ldots,j_{m-1}) \in J$. Пусть $L(h) = \{(j_0,\ldots,j_{m-1}) | 0 \le j_i < q^s, \sum_i j_i > h\}$, и $L^*(h) = \left\{j = \sum_{i=0}^{m-1} j_i q^{si} | (j_0,\ldots,j_{m-1}) \in L(h)\right\}$. В [110] было показано, что коды $C(q,s,m,L(h))$ и $C(q,sm,1,L^*(h))$ эквивалентны. Таким образом, эти коды могут рассматриваться как ОКК с внутренними и внешними кодами, принадлежащими этому же классу. Более того, внешние коды в этой конструкции могут быть рекурсивно разложены таким же способом. Это означает, что код РБЧХ имеет надкод, эквивалентный ОКК с внешними и внутренними расширенными циклическими кодами. Этот факт аналогичен существованию надкода Рида-Маллера кода РБЧХ, что было использовано выше для доказательства того, что последний код имеет “не очень плохое” множество номеров замороженных символов в случае поляризующего преобразования с ядром Арикана.

В связи с этим предлагается использовать следующую кодовую конструкцию. Пусть $\mathcal{C}'$ — $(2^{\mu m}, k', d)$ код РБЧХ, причем t-ый столбец его проверочной матрицы образован степенями элемента $x_t = \sum_{j=0}^{m-1} \beta_{t_j}\gamma_j$, где $(\gamma_0,\ldots,\gamma_{m-1})$ — базис $\mathbb{F}_{2^{\mu m}}$, рассматриваемого как векторное пространство над $\mathbb{F}_{2^\mu}$, $t = \sum_{j=0}^{m-1} t_j 2^{\mu j}, 0 \le t_j < 2^\mu$, и β_i — i-ый элемент $\mathbb{F}_{2^\mu}$. Вышеописанная конструкция ядра РБЧХ F_{2^μ} соответствует случаю $\beta_0 = 0, \beta_i = \alpha^{i-1}, 1 \le i < 2^\mu$, где α — примитивный элемент $\mathbb{F}_{2^\mu}$. Тогда можно построить полярный подкод кода $\mathcal{C}'$ с использованием поляризующего преобразования $A_m = B_{2^\mu,m} F_{2^\mu}^{\otimes m}$ путем наложения дополнительных ограничений $u_i = 0$ на незамороженные символы, соответствующие подканалам с наивысшими значениями вероятности ошибки P_i.

Данная конструкция требует расчета вероятности ошибки P_i в битовых подканалах. Насколько известно автору, в настоящее время отсутствуют аналитические методы решения этой задачи, кроме как для случая двоичного стирающего канала [118]. В связи с этим



(*а*) Ядро РБЧХ (*б*) Ядро РС

Рис. 3.9. Корректирующая способность полярных (под)кодов с ядрами РБЧХ и Рида-Соломона

остается использовать методы статистического моделирования.

Полученный полярный подкод является примером ОККПС, введенных в разделе 2.1.2. Действительно, рассмотрим подмножество $\mathcal{F}_s = \{j_i \in \mathcal{F} \,|\, \lfloor j_i/l \rfloor = s\}$ множества номеров замороженных символов, соответствующих s-му блоку, $0 \le s < l^{m-1}$, и пусть $V^{(s)}$ — соответствующая $|\mathcal{F}_s| \times l^m$ подматрица V. Не ограничивая общности, будем считать, что $V^{(s)}$ содержит единичную подматрицу в столбцах с номерами из $\mathcal{F}_s$, т.е. $V^{(s)} = (\Delta_s | \underbrace{(\Sigma_s | I)\Pi_s}_{|\mathcal{F}_s| \times l} | 0)$, где Π_s — $l \times l$ перестановочная матрица, и Δ_s, Σ_s — некоторые матрицы. Таким образом, возникает система уравнений

$$u_{sl^{m-1}}^{sl^{m-1}+l-1}((\Sigma_s|I)\Pi_s)^T = u_0^{sl^{m-1}-1}\Delta_s^T.$$

Ее решение имеет вид

$$u_{sl^{m-1}}^{sl^{m-1}+l-1} = v(I|\Sigma_s)\Pi_s + u_0^{sl^{m-1}-1}\Delta_s^T(0|I)\Pi_s,$$

где v — произвольный вектор из $\mathbb{F}_2^{l-|\mathcal{F}_s|}$. Таким образом, вместо последовательного оценивания символов $u_{sl^{m-1}}, \ldots, u_{sl^{m-1}+l-1}$ согласно (2.4), как предполагается в методе ПИ, можно найти их совместно путем декодирования в смежном классе $x_s + \mathbb{C}_s$, где $\mathbb{C}_s$ — код, порождаемый матрицей $(I|\Sigma_s)\Pi_s F_l$, и $x_s = u_0^{sl^{m-1}-1}\Delta_s^T(0|I)\Pi_s F_l$, как показано на рис. 1.1, *б*. Это позволяет повысить корректирующую способность декодера и/или снизить его сложность.

На рис. 3.9, *а* представлены результаты моделирования для полярных (под)кодов с ядрами РБЧХ и Арикана, а также полярного кода с ядром Арикана и CRC-16. Декодирование производилось с помощью последовательного алгоритма, описанного в [120]. Видно, что использование ядра РБЧХ обеспечивает значительный выигрыш по сравнению со случаем ядра Арикана. В обоих случаях конструкция полярных подкодов позволяет существенно повысить

корректирующую способность, в особенности в области высоких отношений сигнал/шум.

3. Ядро Рида-Соломона. Результаты [110] позволяют обобщить конструкцию полярных подкодов кодов РБЧХ на случай ядра Рида-Соломона (РС) над $\mathbb{F}_q$. Ядро РС задается матрицей F_l, где $(F_l)_{i,j} = \beta_j^{l-1-i}$, и β_j — различные элементы $\mathbb{F}_q, l \le q$. В [126] было показано, что при $l \le q$ ядро РС обеспечивает наивысшую возможную скорость поляризации. При этом, однако, полярные коды с ядром РС также имеют недостаточное минимальное расстояние.

Для построения кодов с улучшенной корректирующей способностью, положим $l = q$ и представим $(n = q^m, k', d)$ код РБЧХ длины q^m над $\mathbb{F}_q$, t-ый столбец проверочной матрицы которого образован степенями $x_t = \sum_{j=0}^{m-1} \beta_{t_j}\gamma_j$, где $(\gamma_0, \ldots, \gamma_{m-1})$ — базис поля $\mathbb{F}_{q^m}$, рассматриваемого как векторное пространство над $\mathbb{F}_q$, $t = \sum_{j=0}^{m-1} t_j q^j, 0 \le t_j < q^m$, с помощью системы уравнений (2.3), и введем дополнительно $k' - k$ ограничений $u_i = 0$ на незамороженные символы, соответствующие подканалам $\mathcal{W}_m^{(i)}$ с наивысшими вероятностями ошибки. Для нахождения этих вероятностей может использоваться статистическое моделирование.

Пример 8. *Рассмотрим построение* $(16, 8, 6)$ *полярного подкода над* $\mathbb{F}_4$. 4×4 *ядро РС имеет вид*

$$F_4 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & \beta+1 & \beta \\ 0 & 1 & \beta & \beta+1 \\ 1 & 1 & 1 & 1 \end{pmatrix},$$

где β *— примитивный элемент* $\mathbb{F}_4$. *Проверочная матрица* $(16, 9, 6)$ *протокода РБЧХ имеет вид*

$$H = \begin{pmatrix} 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \\ 0&1&\beta&\beta^2&0&1&\beta&\beta^2&0&1&\beta&\beta^2&0&1&\beta&\beta^2 \\ 0&0&0&0&1&1&1&1&\beta&\beta&\beta&\beta&\beta^2&\beta^2&\beta^2&\beta^2 \\ 0&1&\beta^2&\beta&\beta&\beta^2&1&0&1&0&\beta&\beta^2&\beta^2&\beta&0&1 \\ 0&0&0&0&1&1&1&1&\beta^2&\beta^2&\beta^2&\beta^2&\beta&\beta&\beta&\beta \\ 0&1&1&1&\beta&1&0&\beta&\beta&\beta&1&0&\beta&0&\beta&1 \\ 0&0&0&0&\beta^2&\beta^2&\beta&\beta&\beta^2&\beta&\beta^2&\beta&\beta^2&\beta&\beta&\beta^2 \end{pmatrix}$$

В случае поляризующего преобразования $A_2 = B_{4,2}F_4^{\otimes 2}$ *она задает следующую матрицу огра-*

ничений:

$$V = \begin{pmatrix} 0\,0\,0 & 0 & 0\,0\,1\,0\,0 & \beta^2 & 0\,0\,1\,0\,0\,0 \\ 0\,0\,0 & \beta^2 & 0\,0\,1\,0\,0 & 1 & 0\,0\,0\,0\,0\,0 \\ 0\,0\,0 & 0 & 0\,0\,0\,0\,1 & 0 & 0\,0\,0\,0\,0\,0 \\ 0\,0\,0 & 0 & 1\,0\,0\,0\,0 & 0 & 0\,0\,0\,0\,0\,0 \\ 0\,0\,1 & 0 & 0\,0\,0\,0\,0 & 0 & 0\,0\,0\,0\,0\,0 \\ 0\,1\,0 & 0 & 0\,0\,0\,0\,0 & 0 & 0\,0\,0\,0\,0\,0 \\ 1\,0\,0 & 0 & 0\,0\,0\,0\,0 & 0 & 0\,0\,0\,0\,0\,0 \end{pmatrix}.$$

При передаче двоичного образа выхода этого поляризующего преобразования по каналу с АБГШ с $E_s/N_0 = -1$ *дБ наблюдаются следующие вероятности ошибки на символ в подканалах:* $(\underline{0.74}, \underline{0.7}, \underline{0.55}, 0.27, \underline{0.58}, 0.33, 0.12, 0.02, \underline{0.23}, 0.04, 4\cdot10^{-3}, 2\cdot10^{-4}, \underline{0.03}, 4\cdot10^{-4}, 3\cdot10^{-6}, <10^{-6})$. *Таким образом, предлагается наложить дополнительное ограничение* $u_5 = 0$.

На рис. 3.9, *б* представлены результаты моделирования для случая передачи двоичного образа полярного (под)кода с 4×4 ядром РС над полем $\mathbb{F}_{2^2}$ по каналу с АБГШ и двоичной фазовой модуляцией. Декодирование полярных (под)кодов выполнялось с помощью аналога метода Тала-Варди, в котором для расчета вероятностей символов использовался метод, описанный [168]. Как и в ранее рассмотренных случаях, полярный подкод обеспечивает значительный энергетический выигрыш по сравнению с полярным кодом с тем же ядром.

3.2.1.2. Рандомизированные полярные подкоды

Как было показано выше, полярные подкоды кодов РБЧХ в сочетании со списочным или последовательным алгоритмами декодирования обеспечивают значительный энергетический выигрыш по сравнению с полярными кодами и полярными кодами с CRC. Эксперименты показывают, что при небольших размерах списка абсолютное большинство ошибок декодирования связано с потерей декодером правильного пути на ранних фазах. Для обеспечения декодирования (почти) по максимуму правдоподобия с помощью списочного/последовательного алгоритма приходится использовать исключительно большой размер списка. В связи с этим представляет интерес разработка кодовой конструкции, которая была бы ориентирована на случай декодирования с небольшим размером списка L. Коды, предлагаемые в данном разделе, относятся к классу полярных подкодов в широком смысле. Более конкретно, они являются подкодами некоторых классических полярных кодов. В данном разделе ограничимся рассмотрением случая ядра Арикана, хотя предлагаемый подход может быть использован и в случае иных ядер.

1. Исключение из полярного кода ненулевых слов малого веса Для получения (n, k) полярного подкода $\mathcal{C}$ предлагается построить $(n, k+t)$ классический полярный код $\mathcal{C}'$, называемый *базовым кодом*, и выбрать случайным образом его k-мерное линейное подпространство. Пусть d' — минимальное расстояние кода $\mathcal{C}'$, и пусть w'_i — число кодовых слов веса i в этом коде. Можно показать [16], что при равновероятном выборе линейных подпространств матожидание числа кодовых слов веса s в коде $\mathcal{C}$ равно

$$\mathbf{M}\left[w_s\right] = w'_s \frac{2^k - 1}{2^{k+t} - 1} \approx w'_s 2^{-t}, s > 0. \tag{3.18}$$

Параметр t должен быть выбран так, чтобы ожидаемое число $\mathbf{M}\left[w_{d'}\right]$ кодовых слов веса d' в коде $\mathcal{C}$ было достаточно мало. Если $\mathbf{M}\left[w_{d'}\right] < 1$, то с большой вероятностью минимальное расстояние d кода $\mathcal{C}$ окажется больше d'.

В [27] было показано, что ошибочный коэффициент, т.е. число кодовых слов веса d', в классическом полярном коде $\mathcal{C}'$ длины $n = 2^m$ с множеством замороженных символов $\mathcal{F}'$ равно

$$w'_{d'} = 2^{m-r} \sum_{\substack{g \in [n] \setminus \mathcal{F}' \\ \mathrm{wt}(g) = r}} 2^{|\lambda_g|}, \tag{3.19}$$

где $r = \min_{i \in [n] \setminus \mathcal{F}'} \mathrm{wt}(i)$, λ_g — список номеров нулевых битов в целом m-битном числе g, и $|(i_0, \ldots, i_{m-r-1})| = \sum_{j=0}^{m-r-1} (i_j - j)$. Можно также заметить, что любое кодовое слово кода $\mathcal{C}'$ веса $d' = 2^r$ получается как линейная комбинация строки веса d' матрицы A_m и, возможно, каких-то других строк этой матрицы.

Построение случайного линейного подкода кода $\mathcal{C}'$ эквивалентно выбору таких кодовых слов $c = uA_m \in \mathcal{C}'$, которые удовлетворяют $c\tilde{H}^T = 0$, где $\tilde{H}$ — $t \times n$ матрица, выбираемая случайным образом из множества двоичных матриц полного ранга. Это эквивалентно случайному выбору $t \times n$ матрицы полного ранга $\tilde{V} = \tilde{H} A_m^T$. Таким образом, матрица ограничений для кода $\mathcal{C}$ имеет вид $V = \begin{pmatrix} V' \\ \tilde{V} \end{pmatrix}$, где V' — матрица ограничений для кода $\mathcal{C}'$, состоящая из строк веса 1. Не ограничивая общности, будем считать, что последние ненулевые элементы строк матрицы V расположены в различных столбцах.

Для упрощения реализации, а также снижения вероятности потери декодером Тала-Варди правильного пути на ранних фазах декодирования, предлагается наложить ограничения динамического замораживания (2.3) на такие символы u_i, что номера i являются наименьшими возможными, и все незамороженные символы $u_i : \mathrm{wt}(i) = r$ входят в уравнения (2.3) с большой вероятностью. Т.к. всякое кодовое слово $c_0^{n-1} = u_0^{n-1} A_m$ веса 2^r соответствует вектору u_0^{n-1} с $u_i = 1$ для некоторого $i : \mathrm{wt}(i) = r$, последнее условие обеспечивает исключение большой

части кодовых слов малого веса из $\mathcal{C}'$. Первое условие дает возможность списочному декодеру учесть соответствующие ограничения на как можно более ранних фазах декодирования, снижая таким образом вероятность потери правильного пути. При этом, однако, выражение (3.18) перестает быть точной оценкой для компонент весового спектра получаемого кода.

Будем называть ограничения, получаемые указанным способом, ограничениями динамического замораживания типа А (ОДЗ-А), а соответствующие символы в левой части (2.3) — ДЗС типа А (ДЗС-А).

2. Коды для улучшенного списочного декодирования. Рассмотрим построение кодов, которые обеспечивали бы снижение вероятности потери правильного пути в декодере Тала-Варди. Для простоты ограничимся рассмотрением его упрощенной версии "минимум-сумма", задаваемой (1.21)–(1.22).

Пусть u_0^{n-1} — вектор, соответствующий переданному кодовому слову. Рассмотрим некоторое ошибочное событие декодера ПИ, т.е. случай $\tau(u_j, S_m^{(j)}(u_0^{j-1}, y_0^{n-1})) < 0$ для некоторого $j \notin \mathcal{F}$. В этом случае декодер Тала-Варди должен рассмотреть неправильный вектор v_0^j с $R(v_0^j, y_0^{n-1}) > R(u_0^j, y_0^{n-1})$.

В [31] было отмечено, что классические полярные коды характеризуются достаточно низкой вероятностью ошибки на бит. Действительно, большинство ошибочных символов на выходе канала имеют малые по абсолютной величине ЛОПП. Таким образом, с большой вероятностью знак ЛОПП $S_m^{(i)}(v_0^{i-1}, y_0^{n-1}), i > j$, полученный из (1.22) может быть правильным (т.е. быть согласованным со значением u_i), даже если $v_j \neq u_j$ для каких-то $j < i$. Поэтому декодеру может потребоваться обработать большое число замороженных символов, прежде чем вес неправильного пути v_0^i будет в достаточной мере уменьшен согласно (1.20). Может оказаться так, что вес $R(u_0^i, y_0^{n-1})$ правильного пути u_0^i окажется меньше, чем веса L неправильных путей, что приведет к исключению декодером правильного пути из рассмотрения, т.е. ошибке декодирования.

Для снижения вероятности таких событий предлагается выбрать q символов u_i, соответствующих наиболее надежным подканалам, которые были бы заморожены в классическом полярном коде, и наложить на них ограничения динамического замораживания со случайными коэффициентами. Такие ограничения будем называть ограничениями динамического замораживания типа Б (ОДЗ-Б), а соответствующие символы u_i — ДЗС типа Б (ДЗС-Б).

Рассмотрим некоторый ДЗС-Б u_i. Для неправильного пути v_0^i с большой вероятностью получим $u_i \neq \sum_{j<i} V_{s_i,j} v_j$, что приведет к рассогласованию значения правой части этого выражения и знака $S_m^{(i)}(v_0^{i-1}, y_0^{n-1})$. Это приведет к быстрому уменьшению веса $R(v_0^i, y_0^{n-1})$ многих

неправильных путей v_0^i по мере увеличения i.

3. Укорочение. Классические полярные коды с ядром Арикана имеют длину 2^m. Для практических целей необходимо обеспечить возможность построения кодов произвольной длины n. Одним из возможных способов решения этой задачи является укорочение. Корректирующая способность укороченного кода может существенно зависеть от того, на какие символы производится укорочение. В случае полярных кодов укорочение влияет на вероятность ошибки в подканалах поляризующего преобразования, поэтому конфигурация укорочения и множество замороженных символов, вообще говоря, должны выбираться совместно. Такая задача была рассмотрена в [117]. Однако оптимизационный алгоритм, предложенный в этой работе, весьма сложен. Для простоты реализации ограничимся использованием метода укорочения, предложенного в [189]. Более конкретно, предлагается положить $u_i = 0, n \le i < 2^m$ (ограничения укорочения) и исключить из вектора $u_0^{2^m-1} A_m$ символы с номерами $r_m(i)$, где

$$r_m\left(\sum_{j=0}^{m-1} i_j 2^j\right) = \sum_{j=0}^{m-1} i_{m-1-j} 2^j$$

— функция обращения битов. Анализируя структуру матрицы A_m, можно заметить, что ограничения укорочения приводят к тому, что эти символы всегда равны 0. Укорочение должно быть учтено при расчете вероятности ошибки в подканалах $\mathcal{W}_m^{(i)}$.

4. Предлагаемый алгоритм построения кодов. Ниже представлено формальное описание предлагаемого метода построения рандомизированного (n, k) полярного подкода. Здесь t и q обозначают, соответственно, число ОДЗ-А и ОДЗ-Б.

1. Пусть V — $(2^m - k) \times 2^m$ матрица, первоначально заполненная 0.

2. [**Ограничения укорочения**] Пусть $s = 2^m - n$. Установить $V_{0\ldots s-1, n\ldots 2^m-1}$ равной $s \times s$ единичной матрице.

3. Пусть $\mathcal{F} \subset \{0, \ldots, n-1\}$ — множество $n - k - t$ номеров наименее надежных подканалов $\mathcal{W}_m^{(j)}, 0 \le j < n$, задаваемых поляризующим преобразованием A_m. Пусть $\mathcal{N} = \{0, \ldots, n-1\} \setminus \mathcal{F}$. Пусть $\hat{\mathcal{F}}$ — множество из q элементов $\mathcal{F}$, соответствующих наиболее надежным подканалам, и пусть $\tilde{\mathcal{F}} = \mathcal{F} \setminus \hat{\mathcal{F}}$.

4. Пусть $w = \min_{i \in \mathcal{N}} \mathrm{wt}(i)$ — минимальный вес номеров ДЗС-А.

5. Пусть $Z = \{z_0, \ldots, z_{t-1}\} \subset \mathcal{N}$ — множество t целых чисел, получаемых выбором из $\mathcal{N}$ максимальных чисел веса $w, w+1$, и т.д., до тех пор пока не будут выбраны t чисел.

6. [**ОДЗ-А**] Для j с 0 по $t-1$:

   - Присвоить $V_{s,z_j} := 1$.
   - Записать в $V_{s,i}, i \in \mathcal{N}, i < z_j$ независимые двоичные равновероятные случайные значения.
   - Пусть $s := s+1$.

7. [**Ограничения статического замораживания**] Для i с 0 по $n-k-q-t-1$:

   - Присвоить $V_{s,f_i} := 1$, где f_i — i-ый элемент $\tilde{\mathcal{F}}$.
   - Присвоить $s := s+1$.

8. [**ОДЗ-Б**] Для i с 0 по $q-1$

   - Присвоить $V_{s,f_i} := 1$, где f_i — i-ый элемент $\hat{\mathcal{F}}$.
   - Записать в $V_{s,j}, j \in \mathcal{N}, j < f_i$ независимые двоичные равновероятные случайные значения.
   - Пусть $s := s+1$.

На практике независимые двоичные равновероятные случайные значения в этом алгоритме могут быть получены с помощью некоторого генератора псевдослучайных чисел (ГПСЧ), например, регистра сдвига с линейной обратной связью. Тогда код может быть полностью задан начальным значением этого ГПСЧ, параметрами канала $\mathcal{W}(y|x)$ и числами n, k, t, q. Сложность предложенного алгоритма составляет $O(k(t+q))$ обращений к ГПСЧ. Напомним, что построение матрицы ограничений полярного подкода $(n, \tilde{k}, d)$ кода РБЧХ с $(n-k') \times n$ проверочной матрицей $\tilde{H}$ предполагает применение метода Гаусса к матрице $\tilde{H}A_m^T$, что требует $(n-\tilde{k})^2 n$ операций сложения.

Ввиду отсутствия аналитических методов расчета вероятности ошибки списочного/последовательного декодирования полярных (под)кодов, не представляется возможным указать оптимальные значения параметров t и q. Однако эксперименты показывают, что выбор $t = \min(m, n-k)$, $m = \lceil \log_2 n \rceil$ и

$$q = \max(0, \min(Q-t, n-k-t)), \tag{3.20}$$

где $Q = 64$ приводит к кодам с хорошей корректирующей способностью при декодировании списочным/последовательным алгоритмом с $L \approx 32$. В некоторых случаях можно получить несколько лучшие коды путем тщательного выбора этих параметров.

Таблица 3.4. Ошибочный коэффициент рандомизированных полярных $(1024, 512)$ подкодов

t	w'_{16}	P_{SC}	$\mathbf{M}[w_{16}]$	$q = 0$			$q = 64 - t$		
				$\min w_{16}$	$\max w_{16}$	$\overline{w}_{16}$	$\min w_{16}$	$\max w_{16}$	$\overline{w}_{16}$
1	53440	0.231	26720	27064	27360	27214.9	20056	21472	20866.4
2	54464	0.234	13616	13416	13896	13560.9	9872	10932	10381.5
6	54464	0.261	851	910	1072	988.04	568	753	661.68
7	62656	0.266	489.5	400	517	463.78	247	343	296.26
9	66752	0.270	130.4	183	283	226.78	111	184	145.18
10	66752	0.277	65.2	84	148	109.08	46	112	73.72
11	66752	0.290	32.6	15	67	41.1	14	57	35.3
16	91328	0.314	1.39	0	10	3.76	0	10	2.98

5. Численные результаты В данном разделе представлены результаты моделирования, иллюстрирующие корректирующую способность предложенных кодов в канале с АБГШ и двоичной фазовой модуляцией.

На рис. 3.10 и в таблице 3.4 представлены значения ошибочного коэффициента рандомизированных полярных $(n = 1024, k = 512)$ подкодов, построенных для $E_b/N_0 = 1.5$ дБ, а также вероятности ошибки их декодирования методом последовательного исключения и с помощью последовательного алгоритма с различными значениями L. Для каждого t и q были построены 50 кодов. Для каждого из кодов с помощью алгоритма, предложенного в [45], были найдены их кодовые слова малого веса. Во всех случаях минимальное расстояние кодов было не менее 16. В таблице представлены значения ошибочного коэффициента w'_{16} соответствующего $(n, k + t)$ базового кода, задаваемые (3.19), вероятность ошибки декодирования методом ПИ $P_{SC,}$, а также максимальное, минимальное и среднее значения w_{16} для полученных кодов. Можно заметить, что для малых t и $q = 0$ значение $\overline{w}_{16}$ весьма близко к тому, которое вытекает из (3.18). Однако при $t > 7$ и $q = 0$ средний ошибочный коэффициент полученных кодов оказывается существенно выше, чем $\mathbf{M}[w_{16}]$ для случая равномерного распределения на множестве подкодов базового кода. Это обусловлено тем, что предложенный алгоритм построения кодов не обеспечивает равновероятный выбор k-мерных линейных подпространств $(k+t)$-мерного классического полярного кода $\mathcal{C}'$. Можно заметить, что коды, получаемые при $q > 0$ (которые не являются подкодами базового кода $\mathcal{C}'$) имеют существенно меньшее значение $\overline{w}_{16}$.

В обоих случаях средний ошибочный коэффициент $\overline{w}_{16}$ быстро убывает с t. Однако уве-

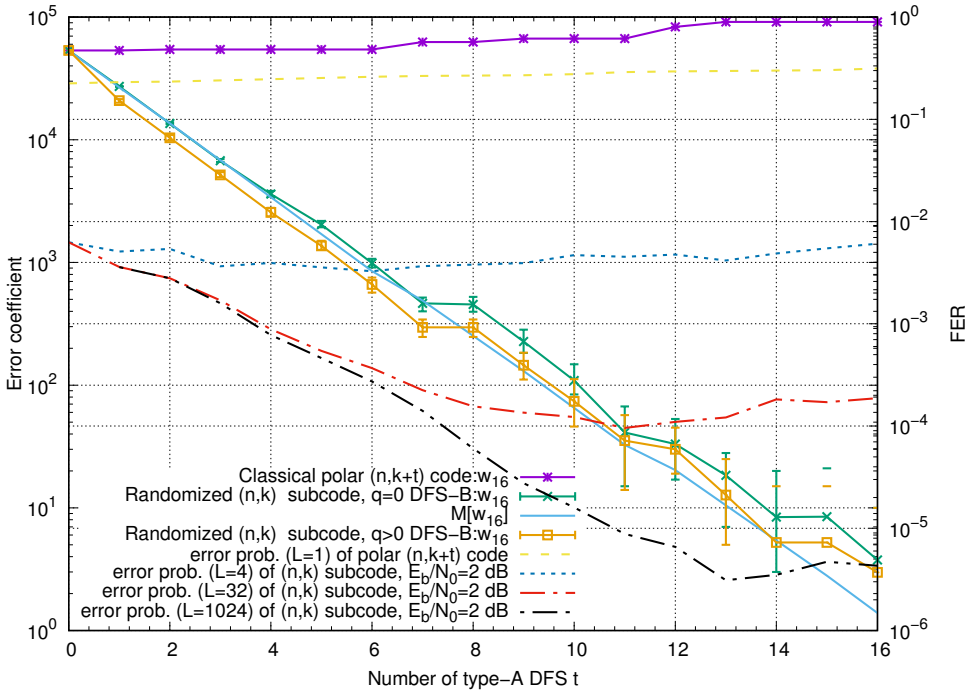


Рис. 3.10. Ошибочный коэффициент w_{16} и вероятность ошибки декодирования рандомизированных полярных $(1024, 512)$ подкодов

личение t требует размораживания символов, соответствующих ненадежным подканалам, что приводит к увеличению P_{SC} и $P(E(L)|\mathbb{C})$, так что с некоторого момента вероятность ошибки списочного декодирования определяется событиями потери правильного пути декодером. Из рис. 3.10 видно, что при различных значениях L, которые определяют среднюю сложность декодирования, минимум вероятности ошибки последовательного алгоритма достигается при различных значениях параметра кодовой конструкции t. Таким образом, предложенный подход позволяет строить коды с учетом требований к сложности декодирования.

На рис. 3.11 представлена корректирующая способность рассмотренных кодов в случае последовательного (см. раздел 4.1) декодирования. Для каждой комбинации t, q был выбран код с $w_{16} \approx \overline{w}_{16}$. Для сравнения, приведены также результаты для $(1024, 512, 24)$ полярного подкода кода РБЧХ и полярного кода с CRC-16. Можно заметить, что коды с $q > 0$ демонстрируют выигрыш по сравнению со случаем $q = 0$, причем наилучшая корректирующая способность в области малых ОСШ достигается при выборе $t = 11, q = 53$. Для $E_b/N_0 > 2$ дБ наилучшая корректирующая способность достигается при выборе $t = 16, q = 48$, что обусловлено существенно меньшим ошибочным коэффициентом этого кода. При $E_b/N_0 \leq 2.1$ дБ рандомизированный полярный подкод с $t = 11, q = 53$ обеспечивает энергетический выигрыш до 0.1 дБ по сравнению с подкодом кода РБЧХ, и до 0.2 дБ по сравнению с полярным кодом с CRC-16. Заметим, что код, получаемый при $t = 16, q = 0$ также обеспечивает выигрыш по

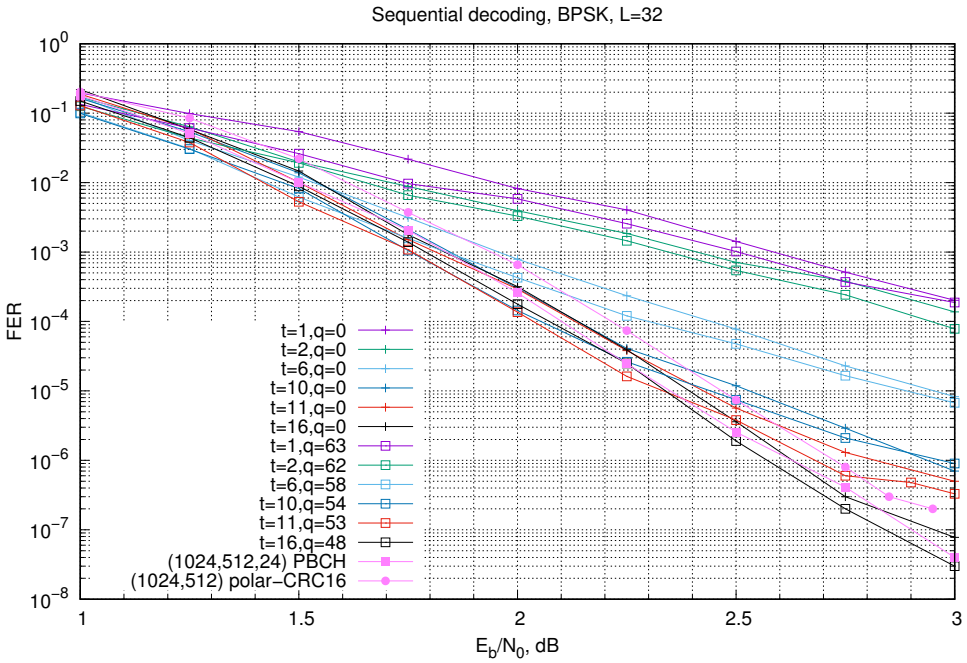
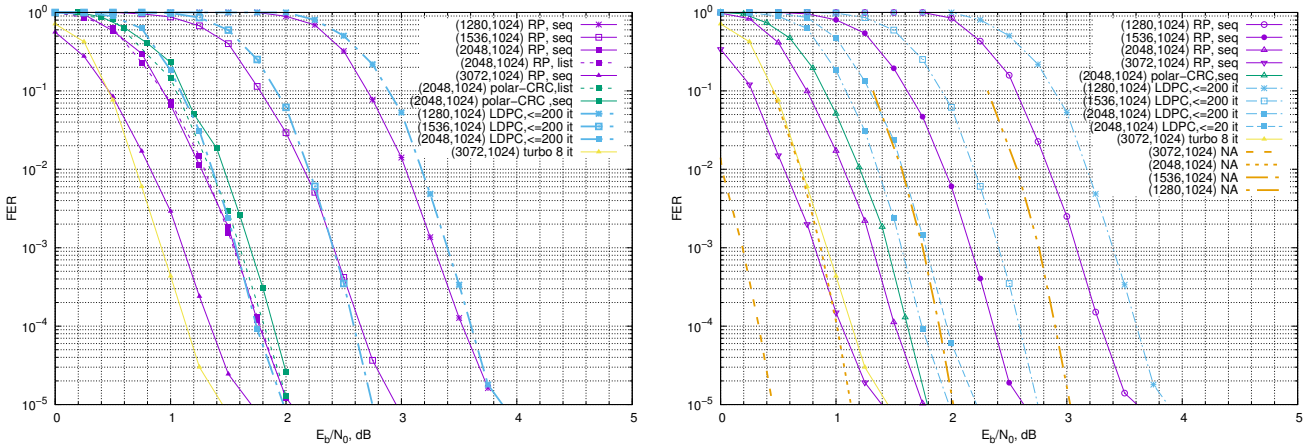


Рис. 3.11. Корректирующая способность рандомизированных полярных $(1024, 512)$ подкодов

сравнению с полярным кодом с CRC. Последний может рассматриваться как особый случай предложенной конструкции с $q = 0$ и $Z = \{n - c, \ldots, n - 1\}$, где $c = 16$— число проверочных символов CRC. Выигрыш обусловлен тем, что предлагаемый подход позволяет декодеру обработать ограничения динамического замораживания на ранних фазах, что снижает вероятность потери декодером правильного пути.

На рис. 3.12 и 3.13 представлена корректирующая способность полярных подкодов с размерностями $k = 1024$ и $k = 4096$. Приведены результаты для рандомизированных полярных подкодов (RP), полярных подкодов кодов РБЧХ (PBCH) и полярного кода с CRC-16. Для сравнения приведены результаты для кодов AR4JA LDPC [161], декодируемых алгоритмом распространения доверия с чередованием [195] и турбо-кода LTE. Для некоторых кодов представлены результаты как в случае последовательного (seq), так и списочного (list) декодирования. Кроме того, представлены значения нормальной аппроксимации (NA) нижней границы вероятности ошибки декодирования [141]. Можно заметить, что при $k = 1024$ и $L = 32$ предложенные рандомизированные полярные подкоды демонстрируют выигрыш по сравнению с полярным кодом с CRC и обеспечивают корректирующую способность, сопоставимую с LDPC кодами. При $L = 512$ предложенные коды обеспечивают энергетический выигрыш до 0.5 дБ по сравнению с кодами LDPC, турбо- и полярным с CRC. Заметим, что превышение экспериментальных кривых для полярных подкодов над нижней границей вероятности ошибки сокращается с увеличением скорости кода. При $k = 4096$ размер списка L, требуемый



(*а*) $L = 32$ (*б*) $L = 512$

Рис. 3.12. Корректирующая способность кодов с $k = 1024$

для обеспечения выигрыша по сравнению с LDPC кодами, оказывается несколько больше. Причиной является то, что хотя доля посредственных подканалов (т.е. имеющих параметры Бхаттачарьи $0 < a < Z_m^{(i)} < b < 1$ для некоторых фиксированных a, b) убывает с m, число таких подканалов, т.е. число незамороженных символов, которые с большой вероятностью вызывают события $E(L)$, растет с m [85]. Заметим, что полярный код с CRC $(2048, 1024)$ демонстрирует заметное ухудшение корректирующей способности в случае последовательного декодирования. Однако корректирующая способность предложенных рандомизированных полярных подкодов оказывается практически одинаковой как в случае последовательного, так и в случае списочного декодирования. Следует также отметить, что рандомизированные полярные подкоды обеспечивают выигрыш по сравнению с полярными подкодами кодов РБЧХ с большим минимальным расстоянием даже в рассмотренной области высоких отношений сигнал/шум.

На рис. 3.14 представлено среднее число операций, выполняемых при декодировании рассмотренных кодов. Для случая полярных и турбо-кодов[2] декодирование требует лишь операций сложения и сравнения, в то время как для LDPC кодов алгоритм, предложенный в [195], предполагает использование операций сложения и вычисления функции $\log\tanh(\cdot)$. Можно заметить, что рандомизированный полярный $(2048, 1024)$ подкод обеспечивает не только меньшую вероятность ошибки, но и меньшую сложность декодирования по сравнению с полярным кодом с CRC. Следует отметить, что на практике при декодировании LDPC-кодов, как правило, ограничиваются существенно меньшим числом итераций алгоритма распространения доверия. Из рисунков 3.12, *б* и 3.14 видно, что это приводит к существенно-

[2] Рассматривался алгоритм Max-Log-MAP.

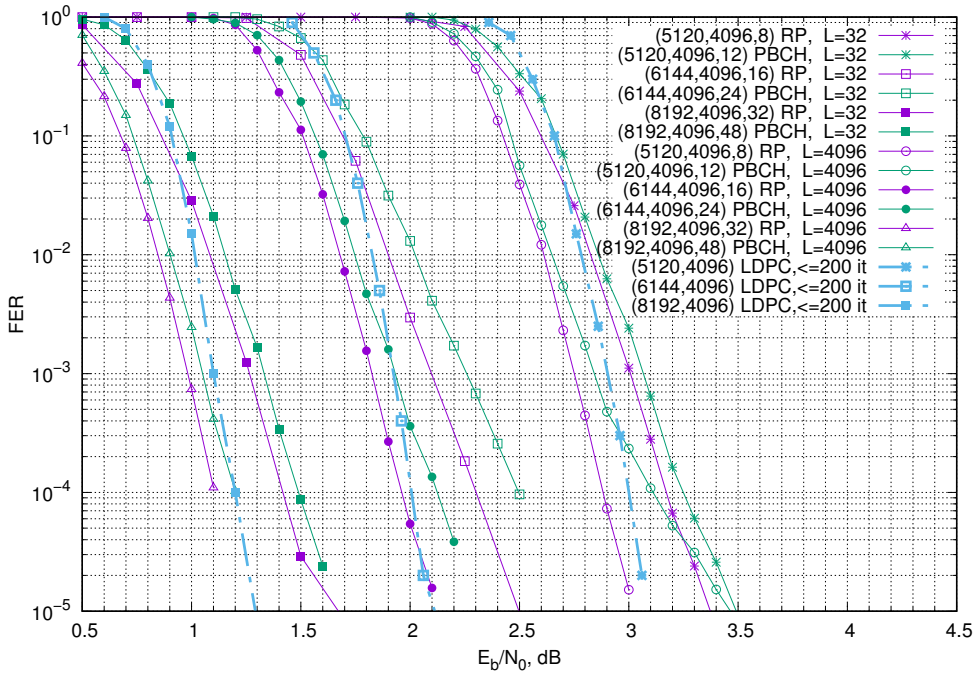


Рис. 3.13. Корректирующая способность кодов с $k = 4096$

му ухудшению корректирующей способности, причем для кода $(2048, 1024)$ при отношениях сигнал/шум более 1.9 дБ сложность последовательного алгоритма декодирования остается меньшей, чем сложность алгоритма декодирования LDPC кода.

На рис. 3.15 представлена зависимость отношения сигнал/шум на символ, требуемого для достижения вероятности ошибки на кодовое слово 10^{-3} в случае двоичной фазовой модуляции для различных значений скорости $R = k/n$ и размерности кода k в случае $q = 0$ и q, задаваемом (3.20). Можно заметить, что использование ДЗС-Б обеспечивает энергетический выигрыш до 0.3 дБ, в особенности для низкоскоростных кодов. Отметим также, что представленные зависимости имеют достаточно гладкий характер, что свидетельствует об эффективности использованного метода укорочения.

На рис. 3.16 представлена корректирующая способность полярных подкодов при их декодировании последовательным алгоритмом с различными значениями числа ДЗС-А q и размера списка L. Видно, что предлагаемый подход позволяет для заданного значения вероятности ошибки декодирования и заданной сложности (определяемой параметром L) подобрать значение t, которое минимизировало бы потребное отношение сигнал/шум.

На рис. 3.17 представлены аналогичные результаты для вероятности ошибки 10^{-2}. Для сравнения приведены результаты для низкоплотностных кодов, предложенных для включения в стандарт 5G [143], декодируемых с помощью алгоритма распространения доверия с 15 итерациями. Видно, что предлагаемые рандомизированные полярные подкоды обеспечивают

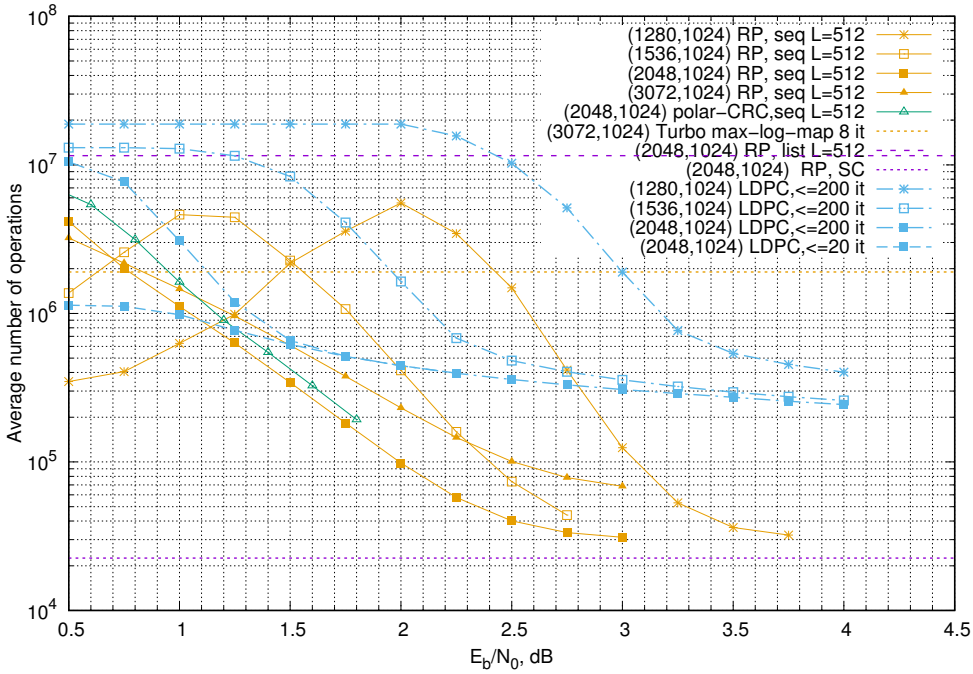


Рис. 3.14. Сложность декодирования кодов с $k = 1024$

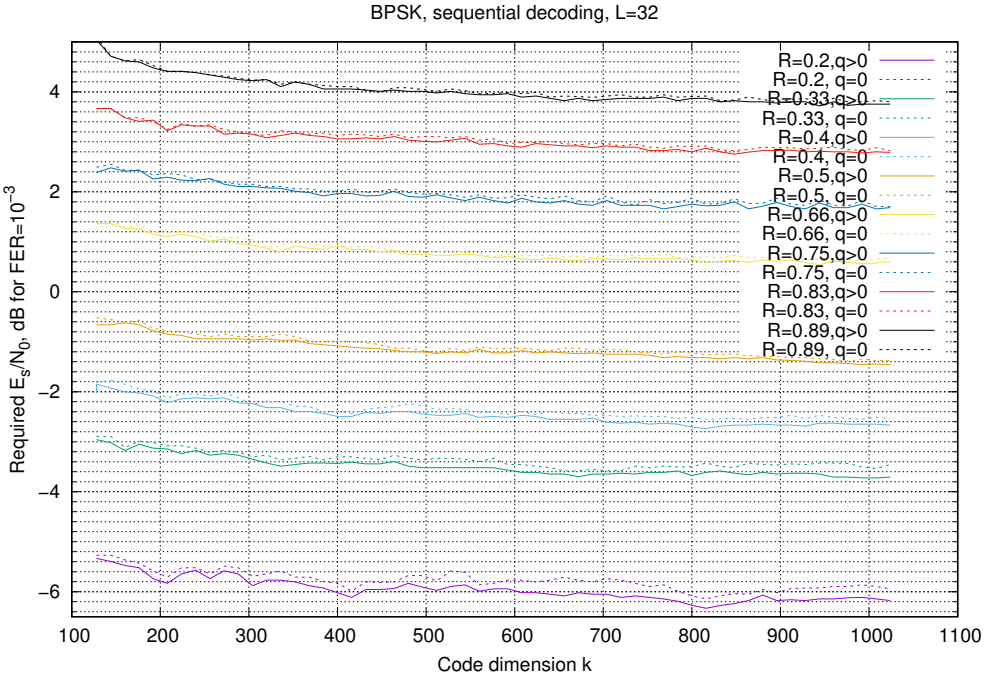


Рис. 3.15. Выигрыш от использования ДЗС-Б

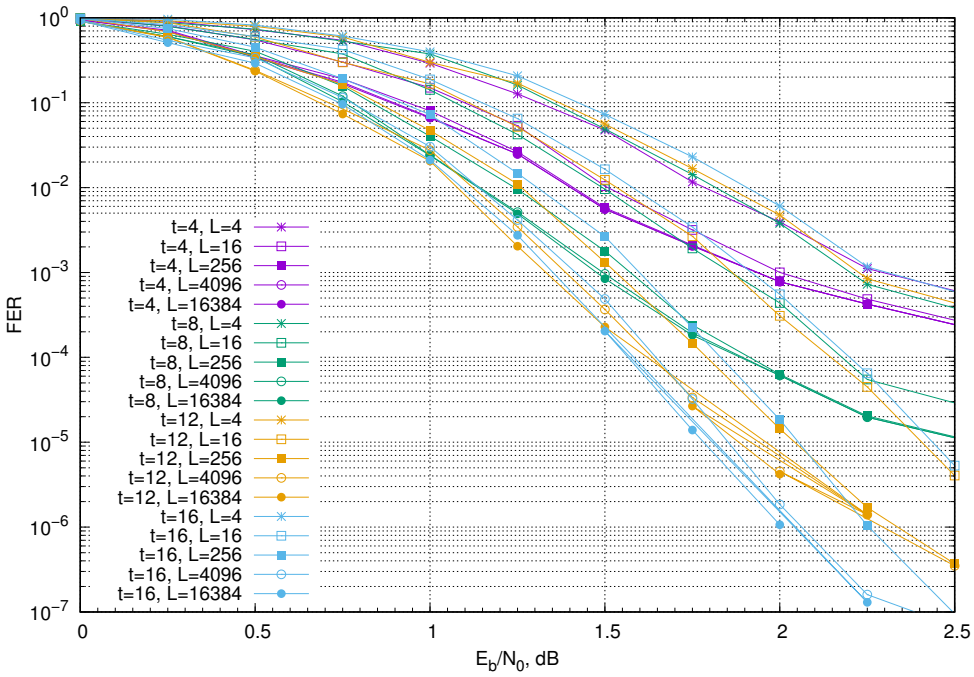


Рис. 3.16. Влияние числа ДЗС-А и размера списка на корректирующую способность рандомизированных полярных подкодов

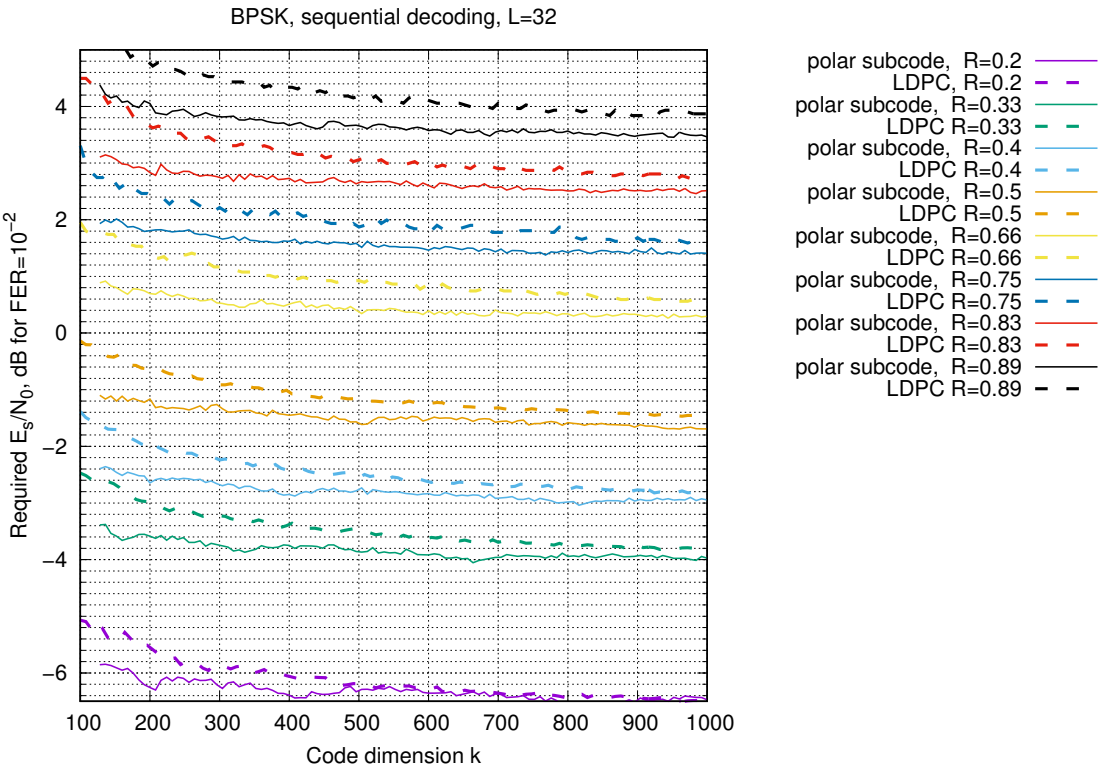


Рис. 3.17. Сравнение с низкоплотностными кодами

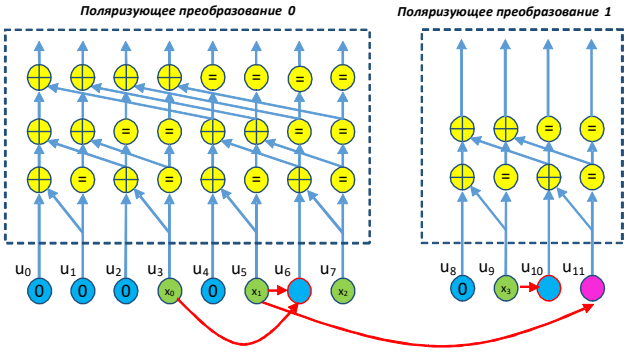


Рис. 3.18. Кодер цепного полярного подкода

существенный выигрыш, в особенности для кодов малых размерностей.

3.2.2. Цепные полярные подкоды

Как было показано выше, с помощью укорочения могут быть получены полярные подкоды произвольной длины. Однако при их декодировании приходится выполнять достаточно большое число бесполезных арифметических операций с ЛОПП, равными ∞. Кроме того, расчет надежности подканалов в случае укороченных кодов может вызвать численные проблемы. В связи с этим представляет интерес разработка обобщения полярных (под)кодов, которое позволило бы непосредственно получить коды произвольной длины.

Рассмотрим построение кода длины $n = \sum_{i=0}^{z-1} 2^{m_i}$, $m_i > m_{i+1}, 0 \le i < z-2$ и размерности k, кодовые слова которого представляли бы конкатенацию выходных векторов поляризующих преобразований A_{m_i}, причем входные вектора этих преобразований имели бы некоторые линейные зависимости. Более конкретно, кодовые слова предлагаемых кодов имеют вид

$$c = xW \underbrace{\begin{pmatrix} A_{m_0} & 0 & \dots & 0 \\ 0 & A_{m_1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_{m_{z-1}} \end{pmatrix}}_{A},$$

где $x \in \mathbb{F}_2^k$ — информационный вектор. Такие коды могут рассматриваться в рамках концепции динамически замороженных символов, введенной в разделе 2.2. На рис. 3.18 представлен пример кодера для предлагаемых кодов. Замороженные символы изображены узлами с синими контурами. Узлы, соответствующие нетривиальным динамически замороженным символам, имеют красный контур, а стрелки показывают линейные зависимости.

3.2.2.1. Методы комбинирования кодов

Рассмотрим некоторые классические методы комбинирования кодов различных длин.

1. Конструкция X4. Пусть $C_i, i \in \{0,,\dots,3\}$, — (n_i, k_i, d_i) коды над $\mathbb{F}_2$ с порождающими матрицами $G^{(i)}$ и проверочными матрицами $H^{(i)}$, причем $C_0 \subset C_1$, $C_2 \subset C_3$, $n_0 = n_1, n_2 = n_3, k_1 - k_0 = k_3 - k_2$ и

$$G^{(1)} = \begin{pmatrix} G^{(0)} \\ G' \end{pmatrix}, \; G^{(3)} = \begin{pmatrix} G^{(2)} \\ G'' \end{pmatrix}.$$

Тогда

$$G = \begin{pmatrix} G^{(0)} & 0 \\ 0 & G^{(2)} \\ G' & G'' \end{pmatrix}$$

является порождающей матрицей $(n_1 + n_2, k_1 + k_2, \min(d_0, d_2, d_1 + d_3))$ кода $\mathcal{C}_{X4}$ [153]. В случае $k_2 = 0$ данная конструкция носит название конструкции X.

2. Конструкция XX. Рассмотрим коды $C_i(n_i, k_i, d_i), 0 \le i \le 5$, где $C_0 = C_1 + C_2, C_3 = C_1 \cap C_2$, $k_4 = k_1 - k_3$ и $k_5 = k_2 - k_3$, и порождающие матрицы имеют вид $G^{(1)} = \begin{pmatrix} G' \\ G^{(3)} \end{pmatrix}, G^{(2)} = \begin{pmatrix} G'' \\ G^{(3)} \end{pmatrix}, G^{(0)} = \begin{pmatrix} G' \\ G'' \\ G^{(3)} \end{pmatrix}$. Тогда матрица

$$G = \begin{pmatrix} G' & G^{(4)} & 0 \\ G'' & 0 & G^{(5)} \\ G^{(3)} & 0 & 0 \end{pmatrix}$$

порождает $(n_0 + n_4 + n_5, k_0, \min(d_3, d_1 + d_4, d_2 + d_5, d_0 + d_4 + d_5)$ код C_{XX} [29]. Данный подход можно рассматривать как двукратное применение конструкции X.

3.2.2.2. Алгебраическая конструкция

Для обеспечения хорошей корректирующей способности минимальное расстояния получаемого таким образом кода должно быть достаточно большим. Этого можно достичь, воспользовавшись конструкциями X4 и XX.

Пусть d — конструктивное расстояние рассматриваемого кода. Выберем $2^{m_i} \times 2^{m_i}$ матрицы $G^{(i)}$, первые κ строк которых порождают $(2^{m_i}, \kappa, d_{i,\kappa})$ коды, $d_{i,\kappa} \ge d_{i,\kappa+1}$. Далее $G^{(i)}$ будут

называться компонентными матрицами. Пусть

$$\delta(i,d) = \max_{\kappa: d_{i,\kappa} < d} d_{i,\kappa}.$$

Не ограничивая общности будем считать, что первые ненулевые элементы строк матрицы $W^{(i)} = G^{(i)} A_{m_i}$ расположены в различных столбцах. Пусть $l_{i,j}$ — позиция первого ненулевого элемента в $W^{(i)}_{j,-}$, где $X_{j,-}$ обозначает j-ый столбец матрицы X.

Будем строить порождающую матрицу (n,k,d) цепного полярного подкода как $G = \begin{pmatrix} \tilde{G} \\ \overline{G} \end{pmatrix}$. Здесь матрицы $\tilde{G}$ и $\overline{G}$ являются блочными, причем блоки имеют 2^{m_i} столбцов. Матрица $\tilde{G}$ является блочно-диагональной с блоками $G(i,d)$, где $G(i,d)$ — матрица, состоящая из некоторых строк $G^{(i)}_{s,-}, 0 \le s < 2^{m_i}$, причем

$$d_{i,s+1} \ge d. \tag{3.21}$$

Матрица $\overline{G}$ содержит ровно два ненулевых блока в каждой строке. Если первый ненулевой блок в строке имеет номер i, тогда он равен некоторой строке $G^{(i)}_{s,-}$, где

$$d_{i,s+1} \ge \delta(i,d). \tag{3.22}$$

Если второй ненулевой блок в той же строке имеет номер $j > i$, то он равен некоторой строке $G^{(j)}_{t,-}$, причем

$$d_{i,t+1} \ge d - \delta(i,d). \tag{3.23}$$

Выбор строк из компонентных матриц осуществляется таким образом, чтобы каждая строка использовалась в матрице G не более одного раза. Это гарантирует, что в результате описанных действий будет получена матрица полного ранга. Правила, определяющие выбор конкретных строк в описанной конструкции, приведены ниже.

Чтобы показать, что минимальное расстояние кода, порождаемого матрицей G, ограничено снизу величиной d, рассмотрим произвольное ненулевое кодовое слово c этого кода. Если c имеет единственный ненулевой блок (например, i-ый), то этот ненулевой блок является кодовым словом $(2^{m_i}, \kappa, d_{i,\kappa})$ кода с $d_{i,\kappa} \ge d$. Если c имеет по крайней мере два ненулевых блока (например, i и $j > i$), то первый ненулевой блок является кодовым словом кода $(n_i = 2^{m_i}, \kappa', d_{i,\kappa'}), d_{\kappa'} \ge \delta(i,d)$, в то время как второй является кодовым словом $(2^{m_j}, \kappa'', d_{\kappa''})$ кода, где $d_{\kappa''} \ge d - \delta(i,d)$.

Пусть $P_{m_i,j}$ — вероятность ошибки в битовом подканале $\mathcal{W}^{(j)}_{m_i}$, задаваемом поляризующим преобразованием A_{m_i}. Для обеспечения возможности эффективного декодирования предлагаемых кодов с помощью списочного/последовательного алгоритма декодирования, должны использоваться следующие дополнительные правила выбора строк матрицы G:

1. k строк $G^{(i)}_{s,-}$, использованные в $\tilde{G}$ или в качестве первых ненулевых блоков матрицы $\overline{G}$, должны соответствовать наименьшим значениям $P_{m_i,l_{i,s}}$, для которых выполняется (3.21) или (3.22), соответственно.

2. Строки $G^{(j)}_{t,-}$, используемые в качестве вторых ненулевых блоков матрицы $\overline{G}$, выбираются из соображений минимизации $P_{m_j,l_{j,t}}$.

3. τ строк $G^{(i)}_{s,-}$ с $d_{i,s+1} \geq d$ и наивысшими значениями $P_{m_i,s}$ используются в качестве первых ненулевых блоков $\overline{G}$. Здесь τ является параметром предлагаемой конструкции.

4. Для любой пары строк матрицы $\overline{G}$, содержащих ненулевые блоки $G^{(i)}_{s,-}$, $G^{(j)}_{t,-}$ и $G^{(i)}_{s',-}$, $G^{(j)}_{t',-}$, из $l_{i,s} < l_{i,s'}$ следует $l_{j,t} < l_{j,t'}$.

Правило 1 гарантирует, что представленный в разделе 4.2.3 обобщенный алгоритм последовательного исключения и его списочный/последовательный аналоги учтут ОДЗ на возможно более ранних фазах декодирования, что позволит снизить вероятность ошибки. Представленные выше правила не претендуют на оптимальность, но эксперименты показывают, что получаемые таким образом коды демонстрируют достаточно хорошую корректирующую способность.

Матрица прекодирования $W = GA$ полученного таким образом цепного полярного (n, k, d) подкода имеет такую же блочную структуру, как и порождающая матрица G. Таким образом, первые ненулевые элементы ее строк расположены в различных столбцах $l_j, 0 \leq j < k$, причем значения l_j однозначно определяются величинами $l_{i,j}$. С целью минимизации вероятности ошибки декодирования представленным ниже обобщенным алгоритмом последовательного исключения, целесообразно использовать такие компонентные матрицы $G^{(i)}$, что соответствующие вероятности ошибки $P_{m_i,l_{i,j}}$ в битовых подканалах были бы как можно меньше. Аналогично конструкции, описанной в разделе 3.2.1.1, воспользуемся такими компонентными матрицами, что их подматрицы порождают коды РБЧХ.

Пример 9. *Построим* $(26, 11, 6)$ *код для двоичного стирающего канала с вероятностью*

стирания 0.5 *и* $\tau = 0$. *Ясно, что* $m_0 = 4, m_1 = 3, m_2 = 1$. *Компонентные матрицы имеют вид*

$$G^{(0)} = \left(\begin{array}{cccccccccccccccc}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\ \hline
1&1&0&0&0&0&1&1&0&0&1&1&1&1&0&0\\
0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0\\
1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0\\
1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0\\ \hline
0&1&0&0&0&0&1&0&1&1&1&0&1&0&0&0\\
0&1&0&1&1&1&0&0&0&1&1&0&0&0&0&0\\ \hline
0&0&0&0&1&1&0&0&1&1&0&0&0&0&0&0\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&1&0&1&0&0&0&0&0&0&0&0&0\\
1&1&0&0&1&1&0&0&0&0&0&0&0&0&0&0\\ \hline
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\ \hline
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{array}\right),$$

$$G^{(1)} = \left(\begin{array}{cccccccc}
1&1&1&1&1&1&1&1\\ \hline
0&0&1&1&1&1&0&0\\
1&1&1&1&0&0&0&0\\
1&0&1&0&1&0&1&0\\ \hline
1&1&0&0&0&0&0&0\\
1&0&1&0&0&0&0&0\\
1&0&0&0&1&0&0&0\\ \hline
1&0&0&0&0&0&0&0
\end{array}\right), G^{(2)} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

Они соответствуют матрицам

$$W^{(0)} = \left(\begin{array}{cccccccccccccccc|c}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1 & 1.5\cdot 10^{-5}\\ \hline
0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&1&0 & 0.037\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0 & 0.014\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0 & 0.0078\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0 & 0.10\\ \hline
0&0&0&1&0&1&1&0&0&1&0&0&0&0&0&0&0 & 0.77\\
0&0&0&0&0&1&0&0&0&1&1&0&1&0&0&0&0 & 0.65\\ \hline
0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0 & 0.47\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0 & 0.23\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0 & 0.53\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 & 0.346\\ \hline
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0 & 0.90\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 & 0.96\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 & 0.985\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 & 0.992\\ \hline
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 & 0.999
\end{array}\right)$$

$$W^{(1)} = \left(\begin{array}{cccccccc|c}
0&0&0&0&0&0&0&1 & 0.0039\\ \hline
0&0&0&0&0&1&1&0 & 0.19\\
0&0&0&0&0&0&1&0 & 0.12\\
0&0&0&1&0&0&0&0 & 0.32\\ \hline
0&0&0&0&1&0&0&0 & 0.68\\
0&0&1&0&0&0&0&0 & 0.81\\
0&1&0&0&0&0&0&0 & 0.88\\ \hline
1&0&0&0&0&0&0&0 & 0.996
\end{array}\right), W^{(2)} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Числа, выписанные около вышеприведенных матриц, являются параметрами Бхаттачарьи подканалов, соответствующих первым ненулевым элементам строк. Для построения порождающей матрицы искомого кода, воспользуемся строками $G^{(0)}_{0,-}, G^{(1)}_{0,-}, G^{(0)}_{3,-}, G^{(0)}_{2,-}, G^{(0)}_{1,-}, G^{(0)}_{4,-}, G^{(1)}_{2,-}, G^{(0)}_{8,-}, G^{(0)}_{10,-}, G^{(0)}_{7,-}, G^{(0)}_{9,-}$ *в качестве первых ненулевых блоков матриц* $\tilde{G}$ *и* $\overline{G}$. *Строка* $G^{(1)}_{2,-}$ *должна быть помещена в* $\overline{G}$ *и дополнена* $G^{(2)}_{0,-}$. *Строки*

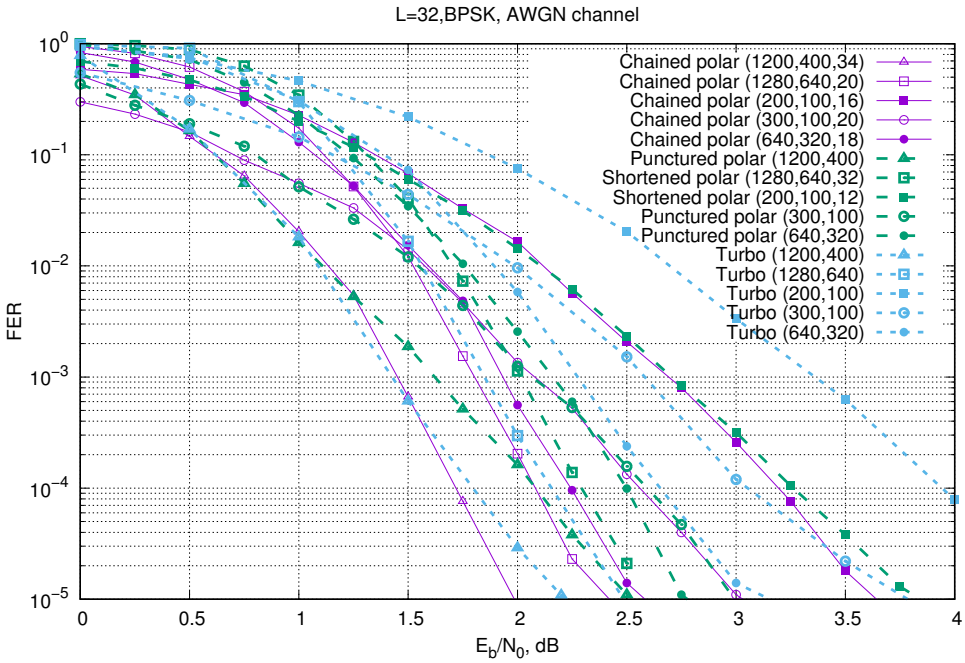


Рис. 3.19. Корректирующая способность цепных полярных подкодов РБЧХ

$G^{(0)}_{9,-}, G^{(0)}_{7,-}, G^{(0)}_{10,-}, G^{(0)}_{8,-}$ *также должны быть помещены в* $\overline{G}$ *и дополнены, соответственно,* $G^{(1)}_{5,-}, G^{(1)}_{3,-}, G^{(1)}_{4,-}, G^{(1)}_{1,-}$. *Таким образом, получим следующую матрицу:*

$$\hat{W} = \left(\begin{array}{cccccccccccccccc|cccccccc|cc}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1 & 0&0&0&0&0&0&0&0 & 0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&1&1&0 & 0&0&0&0&0&0&0&0 & 0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0 & 0&0&0&0&0&0&0&0 & 0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0 & 0&0&0&0&0&0&0&0 & 0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0 & 0&0&0&0&0&0&0&0 & 0&0\\
\hline
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 & 0&0&0&0&0&0&0&1 & 0&0\\
\hline
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 & 0&0&0&0&0&0&1&0 & 0&1\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0 & 0&0&1&0&0&0&0&0 & 0&0\\
0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0 & 0&0&0&1&0&0&0&0 & 0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0 & 0&0&0&0&1&0&0&0 & 0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0 & 0&0&0&0&0&1&1&0 & 0&0
\end{array}\right)$$

Ввиду отсутствия аналитических инструментов для расчета корректирующей способности алгоритмов списочного/последовательного декодирования, параметры d и τ в предлагаемой конструкции приходится выбирать с помощью статистического моделирования.

На рис. 3.19 представлены результаты статистического моделирования для предлагаемых цепных полярных подкодов. Для декодирования использовался алгоритм, описанный в разделе 4.2.2. Для сравнения приведены также результаты для выколотых и укороченных[3] полярных подкодов и турбо-кодов LTE (8 итераций). Можно заметить, что цепные полярные подкоды обеспечивают лучшую корректирующую способность по сравнению с иными рассмотренными кодами. Выигрыш оказывается особенно заметным в области высоких отноше-

[3] Алгоритм поиска оптимальной конфигурации укорочения полярных кодов был предложен в [117]. Он может быть легко обобщен на случай выкалывания.

ний сигнал/шум, что обусловлено как большим минимальным расстоянием цепных кодов, так и возможностью декодера обрабатывать ОДЗ на ранних фазах, что снижает вероятность потери правильного пути декодером.

3.2.2.3. Рандомизированная конструкция

Для практически значимых параметров кода и декодера корректирующая способность цепных полярных подкодов определяется событиями потери правильного пути списочным декодером. В связи с этим в данном разделе представлен аналог вышерассмотренной рандомизированной конструкции укороченных полярных подкодов, который обеспечивает лучшую корректирующую способность в области малых отношений сигнал/шум. Кроме того, процедура построения кодов, предлагаемых в данном разделе, является вычислительно более простой по сравнению с описанной выше алгебраической конструкцией. Предлагаемую процедуру построения удобно описывать в терминах матрицы ограничений V. Следуя методике, описанной в разделе 3.1.1.1, и предполагая, что было передано нулевое кодовое слово, будем считать, что надежность символа $u_i^{(m)}$ характеризуется матожиданием $\mathcal{L}_m^{(i)}$ ЛОПП на выходе соответствующего битового подканала A_m.

1. Удлинение полярного кода на один символ. Рассмотрим построение цепного кода длины $n = 2^{m_0} + 1$, т.е. положим $m_1 = 0, s = 2$. Будем считать, что все символы, соответствующие подканалам $\mathcal{W}_{m_j}^{(i)}$, с надежностью $\mathcal{L}_i^{(m_j)}$ менее некоторого порога T заморожены, $0 \le j < 2, 0 \le i < 2^{m_j}$. Пусть k — размерность соответствующего кода. Но если $u_0^{(m_1)}$ заморожен, иногда можно увеличить размерность кода на единицу. Действительно, в этом случае существует по крайней мере один подканал $\mathcal{W}_{m_0}^{(i)}$ с меньшей надежностью, чем $\mathcal{W}_{m_1}^{(0)}$, поэтому $u_i^{(m_0)}$ также заморожен. Пусть

$$i_0 = \arg \max_{i:\mathcal{L}_i^{(m_0)} < T} \mathcal{L}_{m_0}^{(i)}$$

— номер наиболее надежного подканала, соответствующего замороженному символу $u_{i_0}^{(m_0)}$. Рассмотрим объединенные подканалы

$$\mathcal{W}_{m_0,m_1}^{(i_0,0)}(y_0^{2^{m_0}}, u_0^{i_0-1}|u) = \mathcal{W}_{m_0}^{(i_0)} \boxplus \mathcal{W}_{m_1}^{(0)} = \frac{1}{2} \sum_{u_{i_0}} \mathcal{W}_{m_0}^{(i_0)}(y_0^{2^{m_0}-1}, u_0^{i_0-1}|u_{i_0}) \mathcal{W}_0^{(0)}(y_{2^{m_0}}|u) \tag{3.24}$$

$$\mathcal{W}_{m_0,m_1}^{(i_0,1)}(y_0^{2^{m_0}}, u_0^{i_0}|u) = \mathcal{W}_{m_0}^{(i_0)} \circledast \mathcal{W}_{m_1}^{(0)} = \frac{1}{2} \mathcal{W}_{m_0}^{(i_0)}(y_0^{2^{m_0}-1}, u_0^{i_0-1}|u_{i_0}) \mathcal{W}_0^{(0)}(y_{2^{m_0}}|u) \tag{3.25}$$

Если надежность $\mathcal{L}_{m_0}^{(i_0)} + \mathcal{L}_{m_1}^{(0)}$ подканала $\mathcal{W}_{m_0,m_1}^{(i_0,1)}$ превышает T, по этому подканалу можно передать еще один символ. Соответствующая $(2^{m_0} - k) \times n$ матрица ограничений $\mathbb{V}$ содержит $2^{m_0} - 1 - k$ строк веса 1, и одну строку веса 2, содержащую единицы в позициях i_0 и 2^{m_0}.

Если увеличение размерности кода вышеописанным способом оказалось невозможным, подканал $\mathcal{W}_{m_j}^{(i)}$ может быть использован для того, чтобы повторно передать значение незамороженного символа $u_t^{(m_{1-j})}$ для некоторых $j \in \{0,1\}, 0 \le t < 2^{m_{1-j}}, 0 \le i < 2^{m_j}$. Таким образом можно уменьшить вероятность ошибки в этом символе при декодировании методом последовательного исключения.

Отметим, что метод удлинения линейных кодов, позволяющий нарастить их минимальное расстояние, был представлен в [76]. Но при этом не были описаны эффективные алгоритмы декодирования получаемых таким образом кодов. Предложенный выше подход приводит к кодам, допускающим простое декодирование с помощью обобщенного метода последовательного исключения и его аналогов.

2. Улучшение символов. Предложенный выше подход может быть обобщен на случай произвольных длин следующим образом:

- Коды длины 2^μ будем строить как классические полярные коды.
- Коды длины $2^\mu n$, где $n > 1$ — нечетное число, будем строить как обобщенные каскадные с внутренними полярными кодами Арикана длины 2^μ и внешними кодами длины n, рекурсивно получаемыми с помощью данного метода.
- Коды длины $2^\mu n+1$, где n — нечетное число, будем строить путем применения вышеописанной процедуры удлинения к коду длины $2^\mu n$, рекурсивно полученному с помощью данного метода.

Опишем эту процедуру более подробно. Будем строить матрицу ограничений V таким образом, чтобы символы, передаваемые по недостаточно надежным битовым подканалам некоторого поляризующего преобразования A_{m_j}, повторялись по подканалам одного или нескольких иных преобразований A_{m_p}. Такая операция будет называться улучшением символов. Такой подход позволяет снизить вероятность ошибки декодирования получаемого кода обобщенным алгоритмом последовательного исключения и его списочным/последовательным аналогами. Пусть $u = \left(u^{(m_0)}, \ldots, u^{(m_{s-1})}\right)$ — входной вектор объединенного поляризующего преобразования, задаваемого матрицей A. Будем называть символы $u_i^{(m_j)}$ и $u_q^{(m_p)}$, $m_j > m_p$, смежными, если $\left\lfloor i2^{m_p-m_j} \right\rfloor = q$. Для упрощения построения кода и декодера будем применять операцию усиления только к смежным символам.

Предлагаемая конструкция предполагает, что все символы с надежностью хуже некоторого порога T заморожены. Устанавливая $T = T_0 = 0$ и $T = T_1 = (n+1)2/\sigma^2$, получим коды со

скоростями 1 и 0. Значение $T \in (T_0, T_1)$, обеспечивающее достижение заданной размерности (т.е. числа незамороженных символов) k, может быть найдено итеративно с помощью метода бисекции. Предлагается двухэтапный метод усиления символов. Первый этап, называемый *первоначальным выделением символов*, состоит в поиске незамороженных символов. Будем присваивать переменным $u_i^{m_j}$ состояния $\mathbf{F}, \mathbf{U}, \mathbf{A}$, обозначающие, что соответствующий символ заморожен, разморожен или является вспомогательным. Алгоритм, реализующий первый этап, представлен на рис. 3.20. Представленный на рис. 3.21 алгоритм гарантирует, что значение каждого символа $u_{i_t}^{(m_t)}$, который объявляется замороженным на шаге 14 алгоритма Allocate, передается также по некоторым другим подканалам $\mathcal{W}_{m_p}^{(i_p)}, t \le p \le j$, что обеспечивает снижение вероятности ошибки его приема. Для того, чтобы гарантировать, что эти подканалы не используются для передачи каких-либо других символов, они объявляются "вспомогательными".

Второй этап, называемый *усилением символов*, состоит в построении матрицы ограничений $\mathbb{V}$. Пусть $z_{m_j,i} = i + \sum_{p=0}^{j-1} 2^{m_p}$ — позиция символа $u_i^{(m_j)}, 0 \le i < 2^{m_j}$, в векторе $u = (u^{(m_0)}, \ldots, u^{(m_{s-1})})$. Будем использовать сокращенную запись $u[z_{m_j,i}] = u_i^{(m_j)}$. На рис. 3.21 представлен алгоритм построения матрицы ограничений. Этот алгоритм выявляет замороженные символы, которые могут быть использованы для усиления незамороженных, и формирует ограничения, обеспечивающие передачу таких символов по каналам, соответствующих замороженным символам. В частности, шаг 10 алгоритма усиливает $u_{i_p}^{(m_p)}$ с помощью $u_i^{(m_j)}, p < j$. Далее, выявляются незамороженные символы, которым могут быть усилены замороженными (шаг 16).

Заметим, что шаг 10 алгоритма *Boost* может привести к усилению нескольких незамороженных символов одним и тем же замороженным, как продемонстрировано в примере 10. Точный расчет надежности $\mathcal{L}_{m_p}^{(i_p)}$ получаемых таким образом подканалов требует использования двумерной гауссовской аппроксимации. Для упрощения алгоритма построения кодов и с учетом редкости таких случаев это не учитывается на шаге 10, где используется неточная формула. Значения w_j, вычисляемые вышеприведенным алгоритмом, are равны весу кодовых слов, соответствующих $u_j = 1, u_i = 0, i \ne j, j \notin \mathcal{F}$. Эти значения используются далее для построения ограничений динамического замораживания.

Пример 10. *Воспользуемся вышеописанным методом для построения* $(n = 7, k = 4)$ *кода для канала с АБГШ и среднеквадратическим отклонением шума* $\sigma = 0.93$. *Видно, что* $m_0 = 2, m_1 = 1, m_2 = 0$. *С помощью гауссовской аппроксимации можно вычислить следующие вектора надежностей подканалов* $\mathcal{L}^{(0)} = (0.27, 2.0, 2.75, 9.14), \mathcal{L}^{(1)} = (1.00, 4.57), \mathcal{L}^{(2)} = (2.28)$.

ALLOCATE$(n, k, \mathcal{L}_0^{(0)})$

1 Let $n = \sum_{i=0}^{\zeta-1} 2^{m_i}, m_i > m_{i+1}$
2 $\mathcal{L}_{m_i}^{(2j)} \leftarrow \Xi(\mathcal{L}_{m_i-1}^{(j)}), \mathcal{L}_{m_i}^{(2j+1)} \leftarrow 2\mathcal{L}_{m_i-1}^{(j)}, 0 \le j < 2^{m_i-1}$
3 $T_0 = 0; k_0 = n; T_1 = (n+1)\mathcal{L}_0^{(0)}; k_1 = 0$
4 **while** $k_0 > k_1$
5 **do for** $j = 0, \ldots, \zeta - 1, i = 0, \ldots, 2^{m_j} - 1$
6 **do** $T \leftarrow (T_0 + T_1)/2$
7 **if** $\mathcal{L}_{m_j}^{(i)} \le T$
8 **then** $u_i^{(m_j)} \leftarrow \mathbf{F}$
9 **if** $u_i^{(m_j)} = \mathbf{F}$
10 **then for** $p \leftarrow 0$ **to** j
11 **do** $i_p = \arg \max_{\substack{v: u_v^{(m_p)} = \mathbf{F} \\ \lfloor v 2^{m_j - m_p} \rfloor = i}} \mathcal{L}_{m_p}^{(v)}$
12 $t_0 \leftarrow \max_{\substack{0 \le t < j \\ \sum_{q=t}^{j} \mathcal{L}_{m_q}^{(i_q)} > T}} t$
13 **if** t_0 exists
14 **then** $u_{i_{t_0}}^{(m_{t_0})} \leftarrow \mathbf{U}$
15 $u_{i_p}^{(m_p)} \leftarrow \mathbf{A}, t_0 < p \le j$
16 $K \leftarrow \left| \left\{ (j,i) | u_i^{(m_j)} = \mathbf{U}, 0 \le i < 2^{m_j}, 0 \le j < \zeta \right\} \right|$
17 **if** $K < k$
18 **then** $T_1 \leftarrow T$
19 **else** $T_0 \leftarrow T$
20 $u_i^{(m_j)} \leftarrow \mathbf{F}$ for all $i, j : u_i^{(m_j)} = \mathbf{A}$
21 **return** u_0^{n-1}

Рис. 3.20. Первоначальное выделение символов

Пусть $T = 2.7$.

- *Первоначальная разметка символов:*
  - *Замороженные:* $u_0^{(2)}, u_0^{(0)}$.
  - *Незамороженные:* $u_1^{(2)}, u_2^{(2)}, u_3^{(2)}, u_1^{(1)}$.
  - *Вспомогательные:*$u_0^{(1)}$.
- *Усиление символов:*

BOOST($u_j^{(m_i)}$)

1 $R \leftarrow 0; \mathbb{V} \leftarrow 0_{(n-k)\times n}$
2 **for** $j \leftarrow 0$ **to** $\zeta - 1$
3 **do for** $i \leftarrow 0$ **to** $2^{m_j} - 1$
4 **do** $w_{z_{m_j,i}} \leftarrow 2^{\mathrm{wt}(i)}$
5 **if** $u_i^{(m_j)} = \mathbf{F}$
6 **then** $\mathbb{V}_{R,z_{m_j,i}} \leftarrow 1; Pz_{m_j,i} \leftarrow R$
7 **for** $p \leftarrow 0$ **to** $j - 1$
8 **do** $i_p = \arg \min_{\substack{\lfloor i' 2^{m_j - m_p} \rfloor = i \\ u_{i'}^{(m_p)} = \mathbf{U}}} \mathcal{L}_{m_p}^{(i')}$
9 **if** i_p exists
10 **then** $\mathbb{V}_{R,z_{m_p,i_p}} \leftarrow 1$
11 $\mathcal{L}_{m_p}^{(i_p)} \leftarrow \mathcal{L}_{m_p}^{(i_p)} + \mathcal{L}_{m_j}^{(i)}$
12 $R \leftarrow R + 1$
13 **else for** $p \leftarrow 0$ **to** $j - 1$
14 **do** $i_p = \arg \max_{\substack{\lfloor i' 2^{m_j - m_p} \rfloor = i \\ u_{i'}^{(m_p)} = \mathbf{F}}} \mathcal{L}_{m_p}^{(i')}$
15 **if** i_p exists
16 **then** $\mathbb{V}_{Pz_{m_p,i_p},z_{m_j,i}} \leftarrow 1$
17 $P_{z_{m_j,i}} \leftarrow Pz_{m_p,i_p}$
18 $u_i^{(m_j)} \leftarrow \mathbf{F}; u_{i_p}^{(m_p)} \leftarrow \mathbf{U}$
19 $\mathcal{L}_{m_p}^{(i_p)} \leftarrow \mathcal{L}_{m_p}^{(i_p)} + \mathcal{L}_{m_j}^{(i)}$
20 **return** $\mathbb{V}, w_0^{n-1}$

Рис. 3.21. Усиление символов

– *Усилить* $u_1^{(2)}$ *с помощью* $u_0^{(1)}$: $\mathcal{L}_2^{(1)} = 3.0$

– *Усилить* $u_1^{(1)}$ *и* $u_2^{(2)}$ *с помощью* $u_0^{(0)}$*:* $\mathcal{L}_1^{(1)} = 6.85, \mathcal{L}_2^{(0)} = 5.04$

Таким образом, получим матрицу ограничений

$$V = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Соответствующая проверочная матрица кода имеет вид $H = \mathbb{V}A^T = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$.

Таким образом, получен $(7, 4, 3)$ *код Хемминга.*

Следует отметить, что описанная процедура построения допускает возможность усиления сразу нескольких символов с помощью одного и того же символа, как было показано в примере 10. Сумма величин $\mathcal{L}_{m_j}^{(i)}$, вообще говоря, не является точной характеристикой надежностей таких "усиленных" символов. Однако точный расчет соответствующих надежностей весьма нетривиален, и его реализация привела бы к неоправданному усложнению процедуры построения кодов.

3. Повышение корректирующей способности списочного декодера. Корректирующая способность полярных (под)кодов в случае списочного декодирования в значительной степени зависит от вероятности потери декодером правильного пути на ранних фазах. Для снижения вероятности таких событий предлагается выбрать t замороженных символов с наибольшими значениями $\mathcal{L}_i^{(m_j)}, 0 \le j < s$, и модифицировать соответствующие строки $Pz_{m_j,i}$ матрицы ограничений, записав в элементы $\mathbb{V}_{P(z_{m_j,i}),z_{s,h}}, 0 \le h < i2^{m_s-m_j}, 0 \le s \le j$ двоичные равновероятные независимые (псевдо)случайные числа. Соответствующие символы являются аналогами ДЗС-Б, введенных в разделе 3.2.1.2.

4. Исключение кодовых слов малого веса. Аналогично тому, как это было сделано в разделе 3.2.1.2, рассмотрим построение (псевдо)случайного k-мерного подкода кода размерности $k+t$, построенного с помощью вышеописанного алгоритма. Более конкретно, наложим (псевдо)случайные ОДЗ на символы $u_i^{(m)}$ с номерами малого веса $\mathrm{wt}(i)$, которые участвуют в формировании кодовых слов малого веса. Эти ограничения могут быть сформированы с помощью алгоритма, представленного на рис. 3.22. Данный алгоритм выбирает последний незамороженный символ с номером наименьшего веса и формирует линейное ограничение со случайными коэффициентами, которое включает этот символ, а также другие символы, соответствующие внешним кодам с меньшими либо равными индексами в представлении рассматриваемого кода в виде обобщенного каскадного с перекрестными связями.

Эксперименты показывают, что выбор $t = \log_2 n$ приводит к достаточно хорошим кодам.

5. Численные результаты. На рис. 3.23 представлены результаты статистического моделирования для предложенных рандомизированных цепных полярных подкодов и поляр-

```
SUBCODE(u_0^{n-1}, w_0^{n-1}, s, 𝕍, R)
 1  for ω = 1, 2, 4, ...
 2  do for i = n − 1, ..., 1, 0
 3      do if s = 0
 4            then return
 5            else  if u_i = U ∧ w_i < ω
 6                     then i_0 ← i; P ← 0
 7                          while i_0 ≥ 2^{m_P}
 8                          do i_0 ← i_0 − 2^{m_P}; P ← P + 1
 9                          for p ← 0 to  P
10                          do 𝕍_{R,z_{p,i_0 2^{m_p}}} ← 1
11                              for j ← 0 to  i_0 2^{m_p} − 1
12                              do 𝕍_{R,z_{p,j}} ← RAND()
13                          R ← R + 1; s ← s − 1
14  return 𝕍
```

Рис. 3.22. Построение ограничений динамического замораживания типа A для цепного кода

Таблица 3.5. Конструктивное ОСШ для цепных полярных подкодов

Скорость кода	Конструктивное ОСШ E_b/N_0 $Z(r)$, дБ
1/6	0
1/3	1.5
1/2	1.75
2/3	3.75

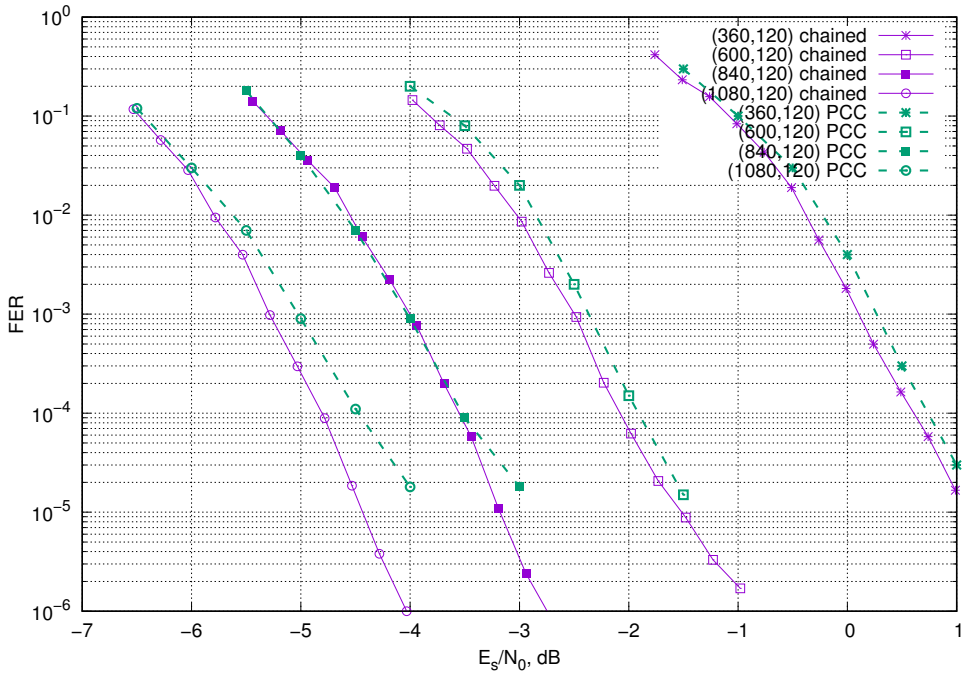


Рис. 3.23. Корректирующая способность рандомизированных цепных полярных подкодов

ных кодов с проверками на четность (PCC) [190]. Можно заметить, что предложенные коды обеспечивают значительный энергетический выигрыш. На рис. 3.24 представлено сравнение корректирующей способности рандомизированных цепных полярных подкодов и различных конструкций укороченных и выколотых кодов [189, 132], а также цепных полярных подкодов, основанных на кодах БЧХ (см. п. 3.2.2.2). Видно, что рандомизированные цепные полярные подкоды обеспечивают значительный энергетический выигрыш.

На рис. 3.25 представлена зависимость отношения сигнал/шум на символ, требуемого для достижения вероятности ошибки 10^{-2} цепными полярными подкодами, от их скорости и размерности. Для сравнения приведены результаты для укороченных кодов, полученных с помощью метода, описанного в разделе 3.2.1.2, и LDPC-кодов, предложенных для включения в стандарт 5G [143], декодируемых с помощью алгоритма распространения доверия с 15 итерациями. Видно, что предлагаемые цепные рандомизированные полярные подкоды обеспечивают примерно такую же корректирующую способность как и укороченные рандомизированные полярные подкоды, в то время как LDPC коды существенно им проигрывают.

3.2.3. Полярные подкоды для систем мобильной связи 5 поколения

Как видно из вышепредставленных результатов, полярные подкоды при фиксированной сложности декодирования особо эффективны для блоков малой длины. В связи с этим они представляют значительный интерес для использования в управляющем канале систем пере-

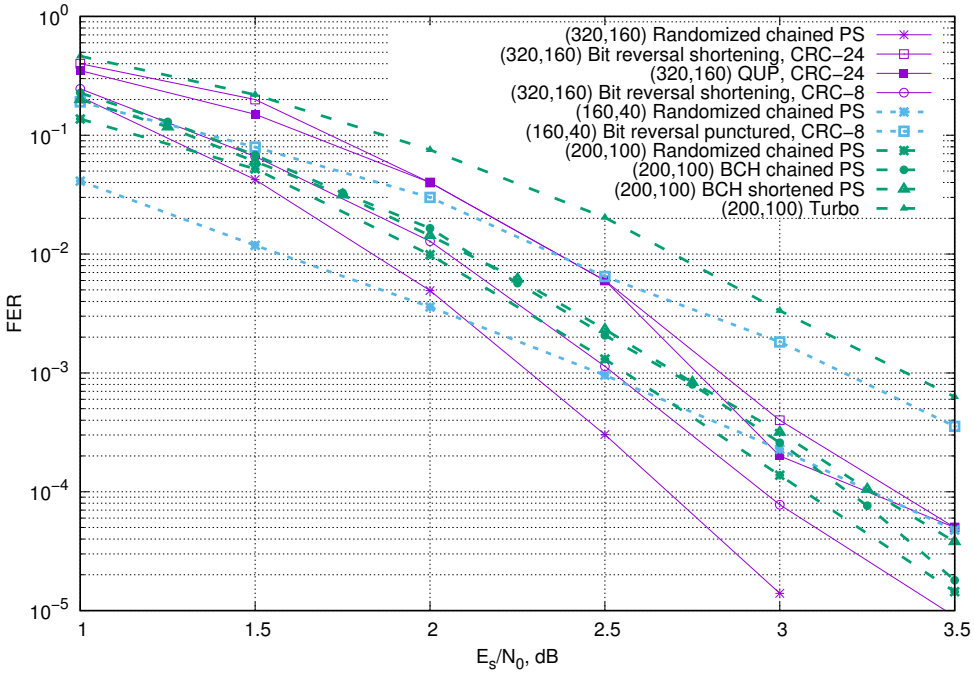


Рис. 3.24. Корректирующая способность цепных полярных подкодов и укороченных кодов иных типов.

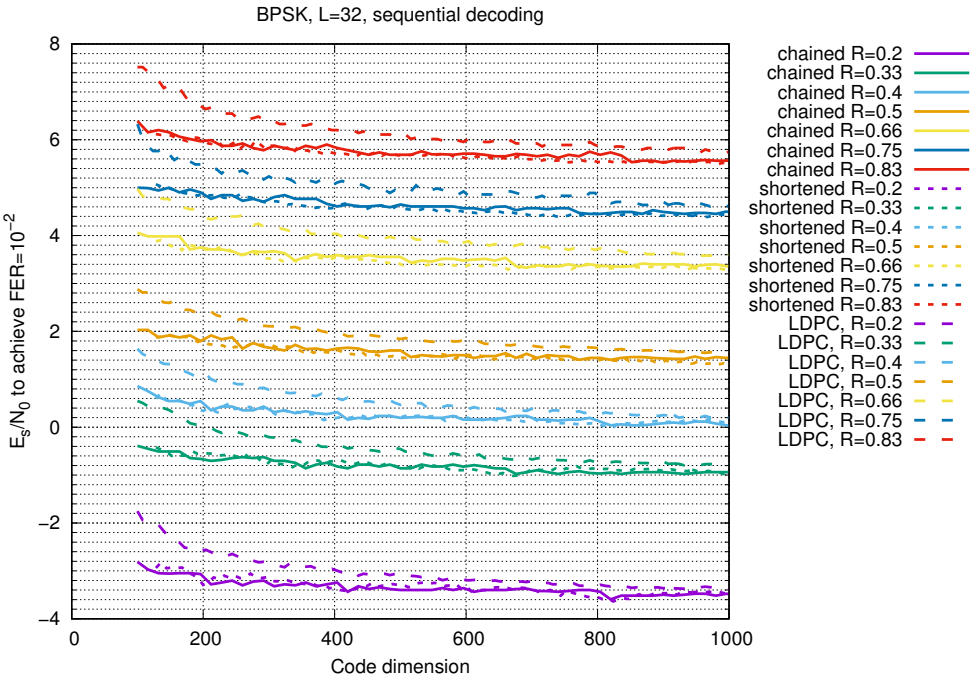


Рис. 3.25. Отношение сигнал/шум, требуемое для достижения вероятности ошибки 10^{-2} цепными полярными подкодами

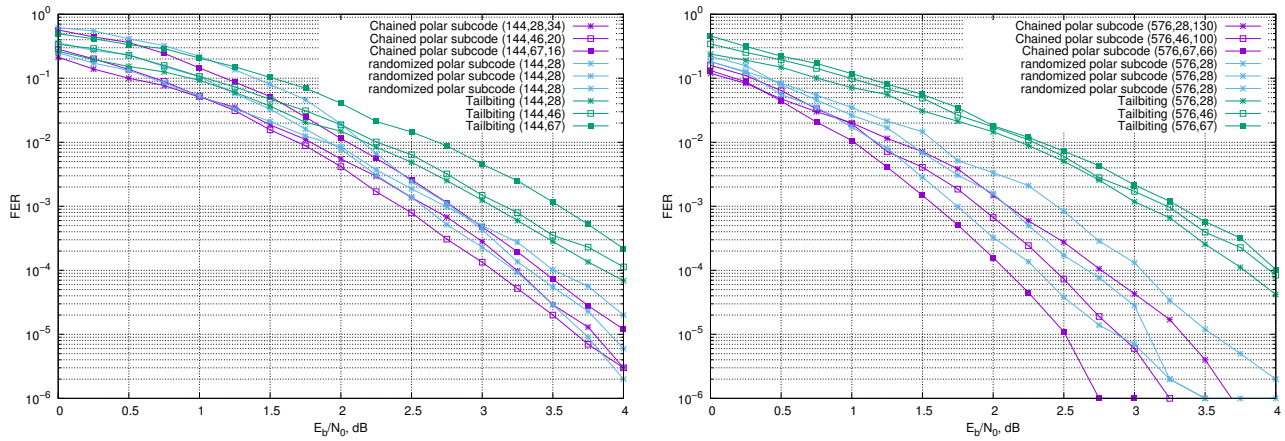


Рис. 3.26. Полярные подкоды и коды с циклическим замыканием

дачи информации. На рис. 3.26 представлено сравнение корректирующей способности полярных подкодов и кодов с циклическим замыканием, используемых в системах мобильной связи 3–4 поколений. Декодирование последних осуществлялось по максимуму правдоподобия. Видно, что полярные подкоды обеспечивают значительный энергетический выигрыш. В связи с этим рандомизированные полярные подкоды были приняты в стандарт мобильной связи 5 поколения [182, п. 5.3.1.2]. Однако ключевым требованием к этим каналам является обеспечение вероятности ложного приема данных не более 10^{-5}. В связи с этим данные должны снабжаться перед кодированием контрольной суммой CRC, которая может использоваться также и для выбора пути из списка, формируемого списочным алгоритмом Тала-Варди. При этом CRC выполняет ту же функцию, что и ДЗС-А, т.е. обеспечивает исключение кодовых слов малого веса из исходного полярного кода путем выбора его псевдослучайного подкода. В связи с этим, а также малой длиной кодируемых блоков данных, общее число динамически замороженных символов ограничено для управляющих каналов величиной $q + t \leq 3$.

3.2.4. Звездные полярные подкоды

Существенным недостатком классических полярных кодов и описанных выше их обобщений является необходимость использования для их декодирования метода последовательного исключения и его аналогов, которые предполагают строго последовательную обработку фаз. В данном разделе будет описана конструкция, основанная на цепных кодах, которая допускает параллельное декодирование принятой последовательности. Предлагаемая конструкция может рассматриваться как развитие идеи звездных решеток [154].

Кодовое слово зведного кода состоит из s кодовых слов коротких полярных (код)кодов, причем входные векторы соответствующих поляризующих преобразований имеют некото-

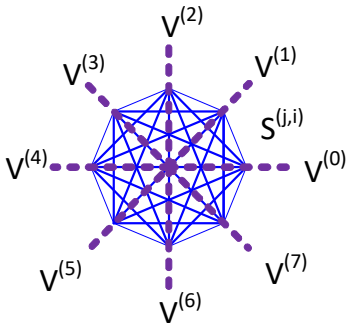


Рис. 3.27. Структура ограничений замораживания звездного кода

рые линейные зависимости. Декодирование таких кодов осуществляется с помощью s экземпляров списочного декодера Тала-Варди (ЭСДТВ) для составляющих кодов. Линейные зависимости используются таким образом, чтобы векторы, найденные ЭСДТВ, могли быть скомбинированы в кодовое слово рассматриваемого звездного полярного подкода.

Ограничимся рассмотрением кодов с ядром Арикана F_2.

3.2.4.1. Общий случай

Определение 9. (n, k) *звездный полярный подкод представляет собой множество векторов*

$$c = (u^{(0)}A^{(0)}, \ldots, u^{(s-1)}A^{(s-1)}), \tag{3.26}$$

где s *— число лучей,* $A^{(i)} = B_{2,m_i}F_2^{\otimes m_i}$, $u^{(i)} \in \mathbb{F}_2^{2^{m_i}}$, *выполняются уравнения*

$$u^{(i)}(V^{(i)})^T = 0, \qquad 0 \le i < s, \tag{3.27}$$

$$u^{(j)}(S^{(j,i)})^T = u^{(i)}(S^{(i,j)})^T, \qquad 0 \le j < i, \tag{3.28}$$

$n = \sum_{i=0}^{s-1} 2^{m_i}$ *— длина кода,* $V^{(i)}$ *—* $\rho_i \times 2^{m_i}$ *матрицы локальных (лучевых) ограничений,* $S^{(i,j)}$ *—* $\mu_{ij} \times 2^{m_i}$ *матрицы перекрестных проверок, причем* $\mu_{ij} = \mu_{ji}$. *Вектор* $u^{(i)}$ *и подвектор* $u^{(i)}A^{(i)}$ *кодового слова называются, соответственно,* i*-ым лучом и блоком.*

Рис. 3.27 иллюстрирует структуру ограничений замораживания звездного полярного подкода. Здесь штриховые линии соответствуют векторам $u^{(i)}$, а непрерывные линии иллюстрируют зависимости между ними.

Пусть $\psi(P, i)$ — номер последнего ненулевого элемента в i-ой строке матрицы P. Пусть $\tau_{i,p} = \psi(V^{(i)}, p), 0 \le p < \rho_i$, и $\sigma_{i,j,t} = \psi(S^{(i,j)}, t), 0 \le t < \mu_{ij}$. Не ограничивая общности, будем считать, что для любых i и j величины $\tau_{i,p}$ и $\sigma_{i,j,t}$ различны, что обеспечивает линейную

независимость векторов коэффициентов уравнений (3.27)–(3.28). Тогда размерность полученного кода равна

$$k = n - \sum_{i=0}^{s-1} \rho_i - \sum_{i<j} \mu_{ij}.$$

Полярные подкоды, рассмотренные в разделе 3.2.1, получаются выбором $s = 1$. В этом случае $\tau_{0,j}$ становятся номерами замороженных символов.

Несистематическое кодирование в звездном коде может быть выполнено следующим образом:

1. Поместить кодируемые данные на позиции с номерами r в векторах $u^{(i)}$, где $r \notin \{\tau_{i,p} | 0 \le p < \rho_i\} \cup \{\sigma_{i,j,t} | j < i, 0 \le t < \mu_{ij}\}$.

2. Для каждого $i \in \{0, \dots, s-1\}$ выполнить следующие действия в порядке возрастания $\tau_{i,p}$ и $\sigma_{i,j,t}$:

   - Вычислить
   $$u^{(i)}_{\tau_{i,p}} = \sum_{r=0}^{\tau_{i,p}-1} u^{(i)}_r V^{(i)}_{pr}, 0 \le p < \rho_i. \tag{3.29}$$
   - Вычислить
   $$u^{(i)}_{\sigma_{i,j,t}} = \sum_{r=0}^{\sigma_{i,j,t}-1} u^{(i)}_r S^{(i,j)} + \sum_{r=0}^{\sigma_{j,i,t}} u^{(j)}_r S^{(j,i)}_{tr}, \tag{3.30}$$
   где $0 \le t < \mu_{ij}$.

3. Вычислить кодовое слов $c = \left(u^{(0)}A^{(0)}, \dots, u^{(s-1)}A^{(s-1)}\right)$.

3.2.4.2. Симметричные звездные полярные подкоды

С целью упрощения описания кодов, а также алгоритмов кодирования и декодирования, предлагается использовать симметричные ограничения перекрестных проверок (3.28). Более конкретно, предлагается построить положить $S^{(0,j)} = S^{(0,1)}, 1 \le j < s$. Тогда кодовые слова симметричного звездного полярного кода имеют вид (3.26), где $\left(u^{(0)}, \dots, u^{(s-1)}\right)\mathbb{V}^T = 0$, и

$$\mathbb{V} = \left(\begin{array}{ccccc} V^{(0)} & 0 & 0 & \dots & 0 \\ 0 & V^{(1)} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & V^{(s-1)} \\ \hline S^{(0,1)} & S^{(1,0)} & 0 & \dots & 0 \\ S^{(0,1)} & 0 & S^{(2,0)} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ S^{(0,1)} & 0 & 0 & \dots & S^{(s-1,0)} \end{array}\right), \tag{3.31}$$

— глобальная матрица ограничений. Для краткости, матрицы $S^{(0,1)}$ и $S^{(i,0)}$ будем обозначать, соответственно, $S^{(0)}$ и $S^{(i)}$.

Описанная глобальная матрица ограничений может быть получена путем многократного применения конструкции X4 [153]. Пусть $H^{(j,i)}$ — проверочные матрицы вложенных $(\nu_i, k_{j,i}, d_{j,i})$ линейных блоковых кодов $C^{(j,i)}$, где $\nu_i = 2^{m_i}$, $j \in \{0,1\}$, $0 \le i < s$, причем $k_{0,i} < k_{1,i}$, $k_{1,i} - k_{0,i} = k_{1,0} - k_{0,0}$, для некоторых матриц $\tilde{H}^{(i)}$ выполняется

$$H^{(0,i)} = \begin{pmatrix} H^{(1,i)} \\ \tilde{H}^{(i)} \end{pmatrix}.$$

Не ограничивая общности, будем считать, что последние ненулевые элементы строк матрицы

$$V^{(0,i)} = H^{(0,i)}(A^{(i)})^T = \begin{pmatrix} V^{(1,i)} \\ \tilde{V}^{(i)} \end{pmatrix}$$

расположены в различных столбцах p. Пусть $\tilde{\mathcal{F}}^{(i)}$ — множество таких чисел p. Предлагается построить матрицы $V^{(i)}$ и $S^{(i)}$ путем дописывания некоторых строк (подробности приведены в разделе 3.2.4.3) к матрицам $V^{(1,i)}$, $\tilde{V}^{(i)}$ соответственно, где $0 \le i < s$.

Лемма 2. *Минимальное расстояние симметричного звездного полярного подкода удовлетворяет*

$$d \ge \min_{0 \le i < s} \left(d_{0,i}, \min_{j \ne i} (d_{1,i} + d_{1,j}) \right).$$

Доказательство. Очевидно, что дописывание любых строк к матрицам $V^{(1,i)}$, $\tilde{V}^{(i)}$ не может уменьшить минимальное расстояние получаемого кода, поэтому эти дополнительные строки могут не учитываться в доказательстве.

Заметим, что часть, соответствующая перекрестным проверкам, глобальной матрицы ограничений (3.31) может быть приведена к виду с $(s-1) \times (s-1)$ блочно-диагональной матрицей, где блоки $S^{(i)}$ занимают любые $s-1$ блочных столбцов из s. Пусть $\mathbb{A}$ — блочно-диагональная матрица с блоками $A^{(i)}$ на диагонали. Можно заметить, что проверочная матрица $\mathbb{H} = \mathbb{V}\mathbb{A}^T$ полученного кода имеет такую же блочную структуру, что и матрица ограничений (3.31).

В связи с этим, не ограничивая общности, ограничимся рассмотрением кодового слова, имеющего ненулевой символ на позициях $0, \dots, 2^{m_0} - 1$, т.е. в блоке 0. Если символы во всех остальных блоках равны нулю, то $u^{(0)}A^{(0)}$ является кодовым словом $C^{(0,0)}$, поэтому его вес не менее $d_{0,0}$. В противном случае, предположим, что имеется ненулевой символ в блоке $j > 0$. Тогда $u^{(0)}A^{(0)}$ и $u^{(j)}A^{(j)}$ являются ненулевыми кодовыми словами $C^{(1,0)}$ и $C^{(1,j)}$, поэтому общее число ненулевых символов в кодовом слове ограничено снизу $d_{1,0} + d_{j,0}$. □

3.2.4.3. Рандомизированная конструкция на основе кодов РБЧХ

На практике корректирующая способность полярных кодов и их обобщений зависит не только от их дистантных свойств, но и от вероятности потери правильного пути на ранних фазах списочного алгоритма Тала-Варди и его обобщений. К сожалению, в настоящее время отсутствуют аналитические инструменты для оценки и контроля данного фактора. В связи с этим предлагается эвристический метод построения кодов, которые допускали бы эффективное декодирование с небольшим размером списка. Суть предлагаемого подхода состоит в наложении псевдослучайных перекрестных проверок (3.30) на символы с достаточно малыми номерами $\sigma_{i,j,t}$. Для неправильных путей в декодере несколько значений нескольких символов $u_r^{(i)}$ должны отличаться от своих истинных значений. Для таких путей условие (3.30) с достаточно большой вероятностью приведет к значительному уменьшению метрики пути в списочном декодер, что в конечном итоге приведет к его исключению. Для обеспечения достаточно большого минимального расстояния у конструируемых кодов, предлагается помимо псевдослучайных ограничений использовать ограничения, получаемые из конструкции X4.

Более конкретно, предлагается следующий метод построения симметричных звездных полярных подкодов. В предлагаемой конструкции каждая матрица $V^{(i)}$ получается путем дописывания к матрице $V^{(1,i)}$ различных строк веса 1, содержащих 1 на позициях $t \notin \tilde{\mathcal{F}}^{(i)}$, соответствующих наименее надежным подканалам (т.е. имеющих наибольшую вероятность ошибки) $\mathcal{W}_{m_i}^{(t)}$. Пусть $\hat{\mathcal{F}}^{(i)}$ — множество таких целых чисел t и $f = \sum_{i=0}^{s-1} |\hat{\mathcal{F}}^{(i)}|$ — общее число таких строк, дописанных к различным матрицам $V^{(i)}$. Матрицы $S^{(i)}$ предлагается формировать путем дописывания к $\tilde{V}^{(i)}$ χ строк, содержащих 1 в различных позициях $p \notin \hat{F}^{(i)} \cup \tilde{F}^{(i)}$, соответствующих наименее надежным подканалам $\mathcal{W}_{m_i}^{(p)}$, и *псевдослучайные* двоичные значения, равномерно распределенные на $\{0,1\}$, в позициях $h < p$. Таким образом, выполняется $\mu_{0,1} = k_{1,0} - k_{0,0} + \chi$, и размерность получаемого кода удовлетворяет

$$k = n - \sum_{i=0}^{s-1} r_{1,i} - f - (s-1)(k_{1,0} - k_{0,0} + \chi), \tag{3.32}$$

где $r_{1,i} = \nu_i - k_{1,i}$ — число строк в матрице $H^{(1,i)}$. Псевдослучайные перекрестные проверки позволяют описанному в разделе 4.4 алгоритму декодирования быстро выявлять и "штрафовать" (т.е. присваивать низкие веса) неверные пути, снижая таким образом вероятность потери правильного пути на ранних фазах декодирования.

Следуя положениям раздела 3.2.1.1, предлагается использовать примитивные коды БЧХ в узком смысле в качестве кодов $C^{(j,i)}$. Это гарантирует, что номера $t \in \tilde{\mathcal{F}}^{(i)}$ соответствуют таким битовым подканалам $\mathcal{W}_{m_i}^{(t)}$, что их надежность (т.е. параметр Бхаттачарьи) улучшается

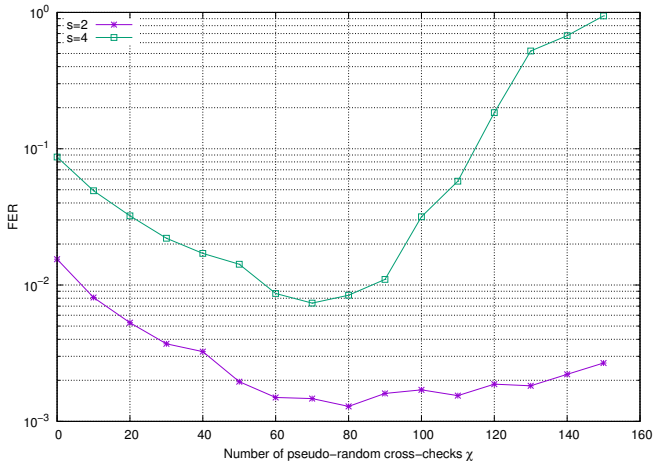


Рис. 3.28. Корректирующая способность $(2048, 1024, 24)$ звездных полярных подкодов с различным числом χ псевдослучайных перекрестных проверок при $E_b/N_0 = 1.5$ дБ

медленно с улучшением надежности исходного канала $\mathcal{W}$. Большинство таких подканалов являются ненадежными.

В случае $m_i = m, 0 \le i < s$, из леммы 2 вытекает, что необходимо положить $d_{0,i} = d, d_{1,i} = d/2$, где d — конструктивное минимальное расстояние кода. Поиск оптимальных значений $d_{r,i}, r \in \{0, 1\}$ в случае лучей различных длин остается открытой проблемой.

Заметим, что нет необходимости сохранять конкретные значения псевдослучайных элементов $S^{(i)}$ в передатчике и приемнике. Достаточно указать алгоритм их генерации и его параметры (например, регистр сдвига с линейной обратной связью, его многочлен связей и начальное значение), что позволит восстановить их при необходимости. Это обеспечивает возможность компактного описания кодов.

3.2.4.4. Численные результаты

Корректирующая способность симметричных звездных полярных подкодов была исследована в случае аддитивного гауссовского канала с двоичной фазовой модуляцией. Был использован алгоритм декодирования, описанный в разделе 4.4. Размер списка при декодировании полагался равным $L = 32$.

Рис. 3.28 иллюстрирует корректирующую способность звездных полярных подкодов с различным числом псевдослучайных перекрестных проверок χ при $E_b/N_0 = 1.5$ дБ. Можно заметить, что для каждого s существует оптимальное значение χ_s числа таких проверок. Если $\chi < \chi_s$, то ЭСДТВ с большой вероятностью удаляют правильные пути до достижения фаз, соответствующих, перекрестным проверкам, полученным из проверочных матриц кодов РБЧХ. Если $\chi > \chi_s$, то получение кода с требуемой размерностью k требует снижения

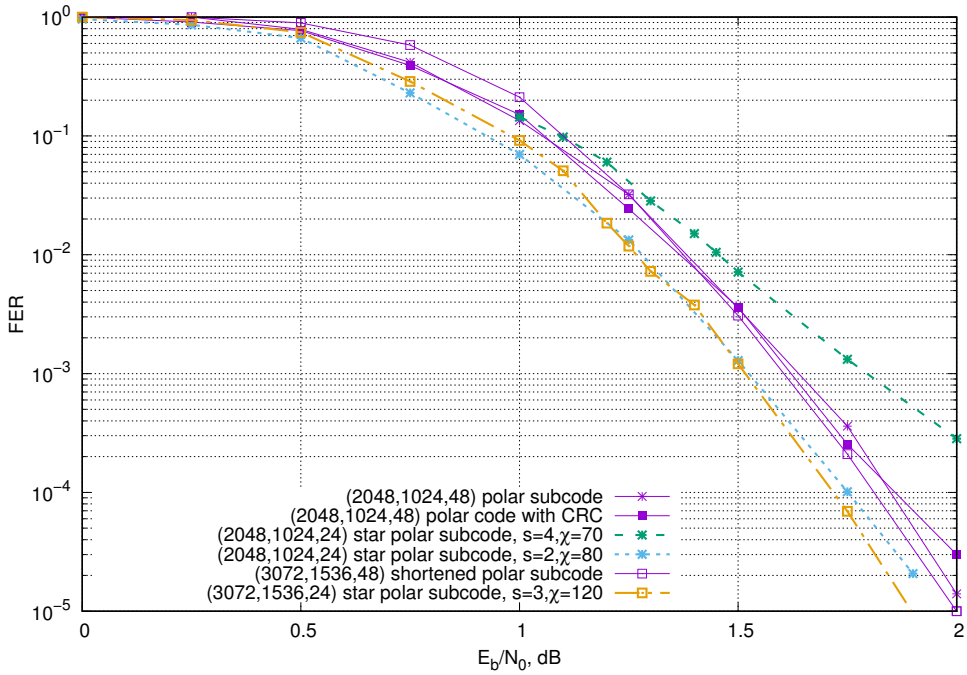


Рис. 3.29. Корректирующая способность звездных полярных подкодов

f в (3.32), т.е. размораживания большого числа символов, соответствующих ненадежным подканалам, что также приводит к повышению вероятности удаления правильного пути на ранних фазах декодирования.

На рис. 3.29 представлено сравнение корректирующей способности звездных полярных подкодов, полярных подкодов кодов РБЧХ [178], т.е. звездных кодов с $s = 1$ лучом, и полярного кода с CRC. Можно заметить, что звездные коды с $s = 2, 3$ демонстрируют выигрыш по сравнению со случаем $s = 1$. Однако при $s = 4$ наблюдается значительное снижение корректирующей способности. При этом, однако, обеспечивается 4-кратное снижение задержки декодирования.

Заметим также, что построение кодов с $s > 1$ потребовало снижения конструктивного минимального расстояния по сравнению с $(2048, 1024, 48)$ полярным подкодом. Согласно (3.32), это необходимо для снижения числа перекрестных проверок, что требуется для обеспечения достаточно большого числа статически замороженных символов f.

При статистическом моделировании во всех рассмотренных сценариях в случае ошибки декодирования возвращаемые декодером кодовые слова были менее правдоподобными по сравнению с переданным кодовым словом, т.е. все ошибки декодирования были обусловлены субоптимальностью используемого алгоритма декодирования.

3.3. Выводы

В данной главе было представлено обобщение полярных кодов, называемое полярными подкодами. Предложенная конструкция основана на предложенном в главе 2 представлении линейных блоковых кодов в виде системы ограничений динамического замораживания. Было показано, что введение небольшого числа динамически замороженных символов позволяет получить коды со значительно лучшими дистантными свойствами по сравнению с классическими полярными кодами, сохранив при этом возможность эффективного декодирования с помощью списочного/последовательного алгоритма ПИ с небольшим размером списка. Полученные таким образом коды демонстрируют существенный энергетический выигрыш по сравнению с известными низкоплотностными и турбо-кодами и допускают возможность учета ограничений на сложность декодирования на этапе построения кода. Предложенная конструкция была описана для случая ядер Арикана, РБЧХ и Рида-Соломона. Представлены как алгебраический, так и рандомизированный методы построения полярных подкодов. Для получения кодов произвольной длины предложены как процедура укорочения кодов с одним поляризующим преобразованием, так и конструкция цепных полярных подкодов, использующих несколько поляризующих преобразований различной размерности.

Полученные коды также демонстрируют значительно лучшую корректирующую способность по сравнению с известными аналогами. В частности, они обеспечивают энергетический выигрыш до 1.75 дБ по сравнению с кодами с циклическим замыканием, применяемыми в системах мобильной связи 3–4 поколений. В связи с этим полярные подкоды приняты к использованию в системах мобильной связи 5 поколения.

Была также предложена конструкция звездных полярных подкодов, допускающая параллельное декодирование, что может быть использовано для снижения его задержки по сравнению с методом ПИ и его аналогами.

Были предложены упрощенные методы расчета надежности битовых подканалов, задаваемых поляризующим преобразованием Арикана, для случая аддитивного гауссовского и релеевского каналов.

Глава 4

Декодирование многочленных кодов

В данной главе рассматривается задача декодирования полярных подкодов и иных многочленных кодов. Предложены обобщения метода последовательного исключения, использование которых в сочетании с представлением декодируемого кода в виде системы ограничений динамического замораживания, введенного в главе 2, позволяет существенно упростить декодирование по сравнению с известными альтернативными подходами.

В разделе 4.1 представлен последовательный алгоритм декодирования полярных кодов и их аналогов. Вопросы декодирования цепных полярных подкодов рассмотрены в разделе 4.2. Алгоритм декодирования расширенного кода Голея, представленного в виде многочленного кода, описан в разделе 4.3. Метод декодирования звездных полярных подкодов приведен в разделе 4.4. В разделе 4.5 описано применение последовательного алгоритма декодирования к кодам Рида-Соломона, представленным в виде ОДЗ.

4.1. Последовательный алгоритм декодирования

В данном разделе представлено описание быстрого алгоритма декодирования полярных (под)кодов с ядром Арикана.

4.1.1. Общее описание

Пусть u_0^{n-1} — входной вектор поляризующего преобразования $A_m = B_{2,m}F_2^{\otimes m}$, использованный передатчиком. Пусть y_0^{n-1} — вектор, полученный в результате передачи кодового слова $c_0^{n-1} = u_0^{n-1}A_m$ по симметричному каналу без памяти. Предлагаемый алгоритм последовательно строит несколько частичных векторов-кандидатов $v_0^{\phi-1} \in \mathbb{F}_2^{\phi}, \phi \leq n$, оценивает, насколько их продолжения v_0^{n-1} могут быть близки к принятой последовательности, и формирует единственное кодовое слово, являющееся решением задачи декодирования.

Множество входных векторов удобно представлять в виде дерева. Узлы этого дерева соответствуют векторам $v_0^{\phi-1}, 0 \leq \phi < n$, удовлетворяющим (2.3). На уровне ϕ каждый узел $v_0^{\phi-1}$ имеет два потомка $v_0^{\phi-1}.0$ и $v_0^{\phi-1}.1$. Корень дерева соответствует пустому вектору. В дальнейшем будет удобно использовать обозначение $v_0^{\phi-1}$ не только для узла дерева, но и для пути от корня к нему, а также считать слова “путь” и “узел” взаимозаменяемыми. Алгоритм декодирования полярного (под)кода должен рассматривать только действительные пути, т.е.

пути, удовлетворяющие (2.3).

Стековый алгоритм декодирования [14, 92, 131, 119] использует приоритетную очередь (ПО)[1] для хранения путей и их весов. ПО является структурой данных, хранящей пары $(M, v_0^{\phi-1})$, где $M = M(v_0^{\phi-1}, y_0^{n-1})$ — вес пути $v_0^{\phi-1}$, и предоставляющей эффективные алгоритмы для следующих операций [89]:

- поместить пару в ПО;
- извлечь пару $(M, v_0^{\phi-1})$ (или только $v_0^{\phi-1}$) с наибольшим значением M;
- удалить заданную пару из ПО.

Будем считать, что ПО способна хранить не более Θ элементов.

В контексте полярных кодов стековый алгоритм декодирования работает следующим образом:

1. Поместить в ПО корень дерева с весом 0. Пусть $t_0^{n-1} = 0$. В дальнейшем вектор t_0^{n-1} используется для числа проходов декодера через различные фазы.
2. Извлечь из ПО узел $v_0^{\phi-1}$ с наибольшим весом. Пусть $t_\phi \leftarrow t_\phi + 1$.
3. Если $\phi = n$, возвратить кодовое слово $v_0^{n-1} A_m$ и завершить работу.
4. Если число допустимых потомков узла $v_0^{\phi-1}$ превышает количество свободных ячеек в ПО, удалить из нее элемент с наименьшим весом.
5. Вычислить веса $M(v_0^{\phi}, y_0^{n-1})$ допустимых потомков v_0^{ϕ} извлеченного узла и поместить их в ПО.
6. Если $t_\phi \geq L$, удалить из ПО все узлы $v_0^{j-1}, j \leq \phi$.
7. Перейти к шагу 2.

В дальнейшем под итерацией будет пониматься один проход этого алгоритма по шагам 2–7.

С точки зрения корректирующей способности параметр L имеет тот же смысл, что и размер списка в списочном алгоритме Тала-Варди, т.к. это значение является верхней границей для числа путей t_ϕ, рассматриваемых декодером на каждй фазе ϕ. Шаг 6 гарантирует, что

[1] В литературе по последовательному декодированию ПО часто называется стеком. Однако реализация предлагаемого алгоритма основывается на структурах данных Тала-Варди [159], которые используют “настоящие” стеки. В связи с этим, далее будет использоваться стандартная терминология компьютерных наук.

алгоритм завершит свою работу за не более чем Ln итераций. Это является также верхней границей для числа элементов, которые могут быть помещены в ПО. Однако эксперименты показывают, что алгоритм может вполне успешно работать и с намного меньшим размером Θ ПО. Шаг 4 гарантирует, что ПО никогда не переполнится.

Существует множество различных подходов к определению весовой функции для последовательного декодирования. В общем случае она должна обеспечивать осмысленное сравнение путей $v_0^{\phi-1}$ различных длин ϕ. Целью данного раздела является определение такой функции. Предлагаемая весовая функция имеет вид

$$M(v_0^{\phi-1}, y_0^{n-1}) = \underbrace{\sum_{i=0}^{\phi-1} \tau(S_m^{(i)}(v_0^{i-1}|y_0^{n-1}), v_i)}_{R(v_0^{\phi-1}|y_0^{n-1})} - \mathbf{M}_{Y_0^{n-1}}\left[\sum_{i=0}^{\phi-1} \tau(S_m^{(i)}(u_0^{i-1}|Y_0^{n-1}), u_i)\right],$$

где

$$\tau(S, v) = \begin{cases} 0, & \operatorname{sgn}(S) = (-1)^v \\ -|S|, & \text{иначе} — \end{cases}$$

штрафная функция, $S_m^{(i)}(v_0^{i-1}, y_0^{n-1})$ — модифицированные логарифмические отношения правдоподобия, определенные ниже, Y_0^{n-1} — случайные величины, соответствующие выходным символам симметричного канала без памяти $W_{Y|C}$, C — случайная величина, соответствующая его входным символам. Первое слагаемое имеет смысл суммарного штрафа, налагаемого на путь $v_0^{\phi-1}$ за его отклонение от вектора, задаваемого жесткими решениями, принятыми на основе знаков ЛОПП $S_m^{(i)}(v_0^{i-1}|y_0^{n-1})$. Эти отклонения могут быть необходимы или для удовлетворения ограничений замораживания, или для исправления ошибок, допускаемых декодером ПИ[2]. Первое слагаемое в точности совпадает с весовой функцией, используемой в аппроксимации "минимум-сумма"[35] алгоритмов Тала-Варди и Нию-Ченя [131].

Второе слагаемое в этом выражении представляет собой матожидание первого слагаемого в предположении, что $v_0^{\phi-1} = u_0^{\phi-1}$, т.е. в случае, если декодер следует по пути, соответствующему истинному вектору u_0^{n-1}, использованному передатчиком. Снижение сложности декодирования в предлагаемом алгоритме обеспечивается в основном за счет использования второго слагаемого.

Пример 11. *Рассмотрим декодирование классического* $(8, 4)$ *полярного кода с* $\mathcal{F} = \{0, 1, 2, 4\}$, *где в качестве весовой функции используется только* $R(v_0^{\phi-1}|y_0^{n-1})$, *т.е. версию минимум-сумма стекового алгоритма Нию-Ченя. Предположим, что ЛОПП принятых символов равны* $(1.0, 1.1, 1.0, 1.1, 1.1, 1.0, -3.05, 2.95)$.

[2] Напомним, что декодер ПИ принимает решения относительно незамороженных символов на основе знаков соответствующих ЛОПП.

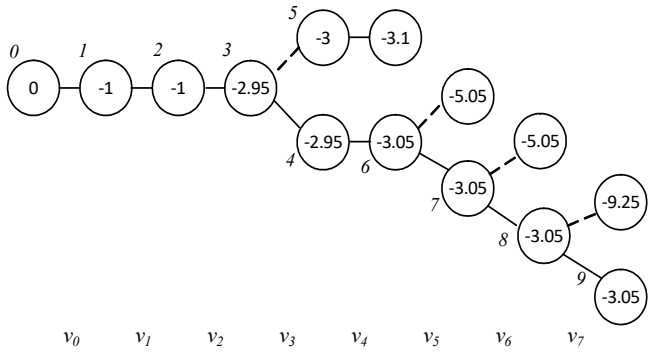


Рис. 4.1. Стековое декодирование полярного кода $(8, 4)$

На рис. 4.1 представлены узлы в кодовом дереве, рассматриваемые данным алгоритмом. Непрерывные и штрихованные линии соответствуют значениям $v_i = 0$ *и* $v_i = 1$*. Числа в кругах показывают веса соответствующих узлов в кодовом дереве. Числа около этих узлов показывают номер итерации стекового алгоритма, на которой соответствующий узел извлекается из ПО.*

Можно заметить, что после итерации 4 декодер переключается на альтернативную ветвь дерева, но после итерации 6 возвращается назад. Было бы разумным считать, что путь $A = (\underline{00000})$*, включающий четыре замороженных символа (подчеркнуты) с весом -3.05 с большей вероятностью соответствует решению задачи декодирования, чем более короткий путь* $B = (\underline{000}1)$*, включающий только три замороженных символа и имеющий незначительно больший вес -3.0. Действительно, для классического полярного кода вес пути с большой вероятностью уменьшается после дополнения его замороженным символом. Таким образом, для снижения сложности декодирования целесообразно ввести весовую функцию таким образом, чтобы путь* B *был исключен из рассмотрения. Такой подход, однако, может привести к увеличению вероятности ошибки декодирования.*

4.1.2. Весовая функция

Корректирующая способность и сложность последовательного декодирования существенным образом зависят от того, каким образом определена весовая функция $M(v_0^{\phi-1}, y_0^{n-1})$. Пусть $U_i, 0 \le i < n$ — случайные величины, соответствующие входным символам поляризующего преобразования. Пусть u_0^{n-1} — вектор входных символов поляризующего преобразования, использованный передатчиком. Классическая метрика Фано для последовательного декодирования сверточных кодов имеет вид

$$P\left\{\mathcal{M}|y_0^{n-1}\right\} = \frac{P\left\{\mathcal{M}, y_0^{n-1}\right\}}{\prod_{i=0}^{n-1} W_Y(y_i)},$$

где $\mathcal{M}$ — сообщение переменной длины (т.е. путь в кодовом дереве), и $W_Y(y_i)$ — вероятностная мера на выходе канала, при условии того, что его входные значения имеют некоторое фиксированное распределение [115]. В контексте полярных кодов непосредственное применение такого подхода соответствовало бы весовой функции

$$M_1(v_0^{\phi-1}, y_0^{n-1}) = \log \mathcal{W}_m^{(\phi-1)}\left\{v_0^{\phi-1}|y_0^{n-1}\right\}.$$

Такая функция была предложена в [131]. Однако этот подход имеет существенные недостатки:

1. Хотя значение веса пути зависят от всех $y_i, 0 \le i < n$, оно не учитывает ограничения замораживания, налагаемые на символы $U_i, i \in \mathcal{F}, i \ge \phi$. В результате могут существовать неправильные пути $v_0^{\phi-1} \ne u_0^{\phi-1}$, имеющие большое число маловероятных продолжений $v_0^{n-1}, v_\phi^{n-1} \in \mathbb{F}_2^{n-\phi}$ с достаточно большой суммарной вероятностью

$$\mathcal{W}_m^{(\phi-1)}\left\{v_0^{\phi-1}|y_0^{n-1}\right\} = \sum_{v_\phi^{n-1}} \mathcal{W}_m^{(n-1)}\left\{v_0^{n-1}|y_0^{n-1}\right\},$$

   что вынуждает стековый декодер исследовать такие пути. Это не является проблемой в случае сверточных кодов, где декодер может восстановиться после пакета ошибок, т.е. найти кодовое слово, которое совпадает с переданным за исключением небольшого числа близко расположенных символов.

2. Ввиду наличия ограничений замораживания, не все векторы $v_0^{\phi-1} \in \mathbb{F}_2^\phi$ соответствуют допустимым путям в кодовом дереве. Это не позволяет должным образом сравнивать пути различных длин, которые включают различное число замороженных символов.

3. Расчет вероятностей $\mathcal{W}_m^{(\phi-1)}\left\{v_0^{\phi-1}|y_0^{n-1}\right\}$ требует использования "дорогих" операций умножения и сопряжен с ошибками округления.

Первая из вышеуказанных проблем может быть решена путем рассмотрения только наиболее вероятного продолжения $v_0^{\phi-1}$, т.е. весовая функция может быть определена как

$$M_2(v_0^{\phi-1}, y_0^{n-1}) = \max_{v_\phi^{n-1} \in \mathbb{F}_2^{n-\phi}} \log \mathcal{W}_m^{(n-1)}\left\{v_0^{n-1}|y_0^{n-1}\right\}. \tag{4.1}$$

Заметим, что максимизация производится по последним $n-\phi$ элементам вектора v_0^{n-1}, в то время как остальные задаются $v_0^{\phi-1}$. Введем также функцию наиболее вероятного продолжения пути

$$\mathbf{V}(v_0^{\phi-1}, y_0^{n-1}) = \arg \max_{v_\phi^{n-1} \in \mathbb{F}_2^{n-\phi}} \log \mathcal{W}_m^{(n-1)}\left\{v_0^{n-1}|y_0^{n-1}\right\},$$

т.е. $M_2(v_0^{\phi-1}, y_0^{n-1}) = \log \mathcal{W}_m^{(n-1)} \left\{ \mathbf{V}(v_0^{\phi-1}, y_0^{n-1}) | y_0^{n-1} \right\}$.

Как будет показано ниже, применение такой функции уже приводит к значительному снижению среднего числа итераций декодера при незначительном ухудшении корректирующей способности. Кроме того, ниже будет показано, что эта функция в точности совпадает с той, которая используется в версии "минимум-сумма" списочного алгоритма Тала-Варди [35], т.е. она может быть вычислена очень простым способом.

Для решения второй проблемы необходимо вычислить вероятности векторов $v_0^{\phi-1}$ с учетом ограничений замораживания. Для этого рассмотрим множество префиксов длины ϕ, удовлетворяющих им, т.е.

$$C(\phi) = \left\{ v_0^{\phi-1} \in \mathbb{F}_2^n \,|\, v_i = \sum_{s=0}^{i-1} V_{j_i,s} v_s, i \in \mathcal{F}, 0 \le i < \phi \right\},$$

где V — матрица ограничений рассматриваемого полярного (под)кода. Определим далее множество наиболее вероятных продолжений этих путей, т.е.

$$\overline{C}(\phi) = \left\{ \mathbf{V}(v_0^{\phi-1}, y_0^{n-1}) | v_0^{\phi-1} \in C(\phi) \right\}.$$

Для любого $v_0^{n-1} \in \overline{C}(\phi)$ вероятность передачи $v_0^{n-1} A_m$, при условии $v_0^{\phi-1} \in C(\phi)$ и получении зашумленного вектора y_0^{n-1} равна

$$\mathbb{W}\left\{ v_0^{n-1} | y_0^{n-1}, C(\phi) \right\} = \frac{\mathcal{W}_m^{(n-1)} \left\{ U_0^{n-1} = v_0^{n-1} | y_0^{n-1} \right\}}{\mathcal{W}_m^{(n-1)} \left\{ U_0^{n-1} \in \overline{C}(\phi) | y_0^{n-1} \right\}}.$$

Идеальная весовая функция могла бы быть определена как

$$\mathbb{M}(v_0^{\phi-1}, y_0^{n-1}) = \log \mathbb{W}\left\{ \mathbf{V}(v_0^{\phi-1}, y_0^{n-1}) | y_0^{n-1}, C(\phi) \right\}.$$

К сожалению, нет простого и очевидного способа расчета $\pi(\phi, y_0^{n-1}) = W_m^{(n-1)} \left\{ U_0^{n-1} \in \overline{C}(\phi) | y_0^{n-1} \right\}$. В связи с этим возникает необходимость построения аппроксимации для этой функции.

Видно, что

$$\pi(\phi, y_0^{n-1}) = \mathcal{W}_m^{(\phi-1)} \left\{ \mathbf{V}(u_0^{\phi-1}) | y_0^{n-1} \right\} + \underbrace{\sum_{\substack{v_0^{\phi-1} \in C(\phi) \\ v_0^{\phi-1} \neq u_0^{\phi-1}}} \mathcal{W}_m^{(\phi-1)} \left\{ \mathbf{V}(v_0^{\phi-1}) | y_0^{n-1} \right\}}_{\mu(u_0^{\phi-1}, y_0^{n-1})}. \tag{4.2}$$

Можно заметить, что $p = \mathbf{M}_{\mathbf{Y}} \left[\frac{\mu(u_0^{n-1}, \mathbf{Y})}{\pi(\phi, \mathbf{Y})} \right]$ есть вероятность того, что версия "минимум-сумма" списочного алгоритма Тала-Варди с бесконечным размером списка не получит $u_0^{\phi-1}$ как наиболее вероятный путь на фазе ϕ. Мы рассматриваем декодирование полярных (под)кодов, которые строятся таким образом, чтобы обеспечить малую вероятность ошибки даже при небольших размерах списка для рассматриваемого канала $W_{Y|C}$. В связи с этим можно считать, что

$p \ll 1$. Из этого следует, что с большой вероятностью $\mu(u_0^{\phi-1}, y_0^{n-1}) \ll \mathcal{W}_m^{(\phi-1)}\left\{\mathbf{V}(u_0^{\phi-1})|y_0^{n-1}\right\}$, т.е. $\pi(\phi, y_0^{n-1}) \approx \mathcal{W}_m^{(\phi-1)}\left\{\mathbf{V}(u_0^{\phi-1})|y_0^{n-1}\right\}$.

Вместе с тем, реальный декодер не может вычислить это значение, т.к. переданный вектор u_0^{n-1} неизвестен на стороне приемника. В связи с этим предлагается дополнительно аппроксимировать логарифм первого члена в (4.2) его матожиданием по $\mathbf{Y}$, т.е.

$$\log \pi(\phi, y_0^{n-1}) \approx \Psi(\phi) = \mathbf{M}_{\mathbf{Y}}\left[\mathcal{W}_m^{(\phi-1)}\left\{\mathbf{V}(u_0^{\phi-1})|\mathbf{Y}\right\}\right]$$

Заметим, что эта величина зависит только от ϕ и канала $W_{Y|C}$, и может быть вычислена заранее.

Таким образом, вместо идеальной весовой функции $\mathbb{M}(v_0^{\phi-1}, y_0^{n-1})$ предлагается использовать приближенную

$$M_3(v_0^{\phi-1}, y_0^{n-1}) = M_2(v_0^{\phi-1}, y_0^{n-1}) - \Psi(\phi). \tag{4.3}$$

4.1.2.1. Эффективное вычисление весовой функции

Рассмотрим вычисление $R_m^{(\phi-1)}(v_0^{\phi-1}, y_0^{n-1}) = M_2(v_0^{\phi-1}, y_0^{n-1})$. Введем модифицированные логарифмические отношения правдоподобия

$$S_m^{(\phi)}(v_0^{\phi-1}|y_0^{n-1}) = R_m^{(\phi)}(v_0^{\phi-1}.0, y_0^{n-1}) - R_m^{(\phi)}(v_0^{\phi-1}.1, y_0^{n-1}). \tag{4.4}$$

Можно заметить, что

$$R_m^{(\phi)}(v_0^{\phi}, y_0^{n-1}) = R_m^{(\phi-1)}(v_0^{\phi-1}, y_0^{n-1}) + \tau(S_m^{(\phi)}(v_0^{\phi-1}|y_0^{n-1}), v_\phi), \tag{4.5}$$

где

$$\tau(S, v) = \begin{cases} 0, & \text{если } \operatorname{sgn}(S) = (-1)^v \\ -|S|, & \text{иначе.} \end{cases}$$

— штрафная функция.

Действительно, пусть $\tilde{v}_0^{n-1} = \mathbf{V}(v_0^{\phi-1})$. Если $v_\phi = \tilde{v}_\phi$, то наиболее вероятные продолжения $v_0^{\phi-1}$ и v_0^{ϕ} совпадают. В противном случае разность логарифмов их вероятностей в точности равна $-\left|S_m^{(\phi)}(v_0^{\phi-1}|y_0^{n-1})\right|$.

Начальное значение рекурсии (4.5) равно

$$R_m^{(-1)}(y_0^{n-1}) = \log \prod_{i=0}^{n-1} \mathcal{W}_0^{(0)}\left\{\hat{c}_i|y_i\right\},$$

где $\hat{c}_i$ — жесткие решения, соответствующие y_i. Однако это значение может быть заменено на 0, т.к. оно не влияет на процесс выбора путей рассматриваемым алгоритмом.

Для того, чтобы получить простую формулу для функции $M_2(v_0^{\phi}, y_0^{n-1}) = R_m^{(\phi)}(v_0^{\phi}, y_0^{n-1})$, заметим, что

$$\mathcal{W}_m^{(n-1)}\left\{v_0^{n-1}|y_0^{n-1}\right\} = \mathcal{W}_{m-1}^{(n/2-1)}\left\{v_{0,e}^{n-1} \oplus v_{0,o}^{n-1}|v_0^{\frac{n}{2}-1}\right\} \cdot \mathcal{W}_{m-1}^{(n/2-1)}\left\{v_{0,o}^{n-1}|y_{\frac{n}{2}}^{n-1}\right\}.$$

Следовательно,

$$R_\lambda^{(2i)}(v_0^{2i}, y_0^{2^\lambda-1}) = \max_{v_{2i+1}}\left(R_{\lambda-1}^{(i)}\left(v_{0,e}^{2i+1} \oplus v_{0,o}^{2i+1}, y_0^{2^{\lambda-1}-1}\right) + R_{\lambda-1}^{(i)}\left(v_{0,o}^{2i+1}, y_{2^{\lambda-1}}^{2^\lambda-1}\right)\right), \tag{4.6}$$

$$R_\lambda^{(2i+1)}(v_0^{2i+1}, y_0^{2^\lambda-1}) = R_{\lambda-1}^{(i)}\left(v_{0,e}^{2i+1} \oplus v_{0,o}^{2i+1}, y_0^{2^{\lambda-1}-1}\right) + R_{\lambda-1}^{(i)}\left(v_{0,o}^{2i+1}, y_{2^{\lambda-1}}^{2^\lambda-1}\right), \tag{4.7}$$

где $0 < \lambda \le m$, и начальные значения для этих рекурсивных выражений задаются $R_0^{(0)}(b, y_j) = \log W_0^{(0)}\{b|y_j\}$, $b \in \{0, 1\}$. Из (4.4) получим, что

$$\begin{aligned} S_\lambda^{(2i)}(v_0^{2i-1}|y_0^{2^\lambda-1}) &= \max(J_0 + K_0, J_1 + K_1) - \max(J_1 + K_0, J_0 + K_1) \\ &= \max(J_0 - J_1 + K_0 - K_1, 0) - \max(K_0 - K_1, J_0 - J_1) \\ S_\lambda^{(2i+1)}(v_0^{2i}|y_0^{2^\lambda-1}) &= J_{v_{2i}} + K_0 - J_{v_{2i}\oplus 1} - K_1 \end{aligned}$$

где $J_c = R_{\lambda-1}^{(i)}((v_{0,e}^{2i-1} \oplus v_{0,o}^{2i-1}).c, y_0^{2^{\lambda-1}-1})$, $K_c = R_{\lambda-1}^{(i)}(v_{0,o}^{2i-1}.c, y_{2^{\lambda-1}}^{2^\lambda-1})$. Заметим, что

$$J_0 - J_1 = a = S_{\lambda-1}(v_{0,e}^{2i-1} \oplus v_{0,o}^{2i-1}|y_0^{2^{\lambda-1}-1})$$

и

$$K_0 - K_1 = b = S_{\lambda-1}^{(i)}(v_{0,o}^{2i-1}|y_{2^{\lambda-1}}^{2^\lambda-1})$$

Из этих выражений можно получить, что модифицированные ЛОПП могут быть вычислены как

$$\begin{aligned} S_\lambda^{(2i)}(v_0^{2i-1}|y_0^{2^\lambda-1}) &= Q(a, b) = \operatorname{sgn}(a)\operatorname{sgn}(b)\min(|a|, |b|), \\ S_\lambda^{(2i+1)}(v_0^{2i}|y_0^{2^\lambda-1}) &= P(v_{2i}, a, b) = (-1)^{v_{2i}}a + b. \end{aligned}$$

Эти формулы совпадают с аппроксимацией “минимум-сумма” для (1.13)–(1.14) [152, 83, 35]. Но, как было показано выше, на самом деле это точные значения, характеризующие вероятность наиболее правдоподобного продолжения пути $v_0^{\phi-1}$ в кодовом дереве.

Следует отметить, что вычисление весовой функции (4.3) требует использования только операций сложения и сравнения. Промежуточные значения, возникающие в вышеприведенных выражениях, могут быть переиспользованы так же, как и в алгоритме Тала-Варди [159]. Т.к. для любого ϕ последовательный алгоритм декодирования извлекает из ПО пути длины ϕ не более L раз, в худшем случае сложность предложенного алгоритма составляет $O(Ln\log n)$. Но, как будет показано в разделе 4.1.5, средняя сложность декодирования оказывается значительно меньшей.

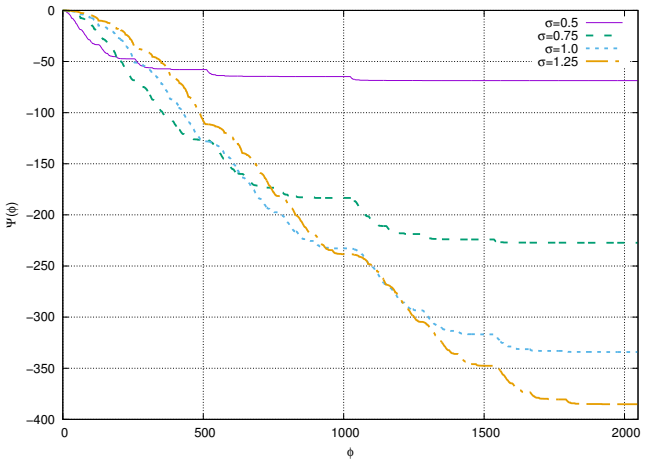


Рис. 4.2. Эвристическая функция для канала с АБГШ, $m = 11$.

4.1.2.2. Эвристическая функция

Функция $\Psi(\phi)$ равна матожиданию модифицированной логарифмической вероятности префикса длины ϕ истинного пути. Предложенная выше весовая функция показывает отклонение пути $v_0^{\phi-1}$ от усредненного поведения правильного пути. Таким образом, эта функция схожа с эвристической функцией, используемой в алгоритме A^* поиска кратчайшего пути [84, 57]. Она может быть построена заранее в предположении о том, что было передано нулевое кодовое слово. Действительно, функции распределения $F_\lambda^{(i)}(x)$ ЛОПП $S_\lambda^{(i)}$ имеют вид [95]

$$F_\lambda^{(2i)}(x) = \begin{cases} 2F_{\lambda-1}^{(i)}(x)(1 - F_{\lambda-1}^{(i)}(-x)), & x < 0 \\ 2F_{\lambda-1}^{(i)}(x) - (F_{\lambda-1}^{(i)}(-x))^2 - (F_{\lambda-1}^{(i)}(x))^2, & x > 0 \end{cases} \tag{4.8}$$

$$F_\lambda^{(2i+1)}(x) = \int_{-\infty}^{\infty} F_{\lambda-1}^{(i)}(x-y) dF_{\lambda-1}^{(i)}(y), \tag{4.9}$$

где $F_0^{(0)}(x)$ — функция распределения ЛОПП на выходе канала. Таким образом, получим

$$\Psi(\phi) = -\sum_{i=0}^{\phi-1} \int_{-\infty}^{0} F_m^{(i)}(x) dx.$$

Эвристическая функция $\Psi(\phi)$ зависит только от m и свойств канала, т.е. она может быть использована в декодере любого кода заданной длины.

На рис. 4.2 представлены графики эвристической функции для случая канала с АБГШ и двоичной фазовой модуляцией для различных значений среднеквадратического отклонения шума σ.

4.1.2.3. Обсуждение

Поясним физический смысл величин $M_2(u_0^{\phi}, y_0^{n-1})$. Пусть S_0^{n-1} — вектор ЛОПП, соответствующих зашумленной последовательности y_0^{n-1}. Пусть

$$E(c_0^{n-1}, S_0^{n-1}) = -\sum_{i=0}^{n-1} \tau(S_i, c_i)$$

— корреляционная невязка (известная также как эллипсоидальный вес) вектора $c_0^{n-1} \in \mathbb{F}_2^n$ [183, 122]. Можно показать, что задача декодирования по максимуму правдоподобия в коде $\mathcal{C}$ для случая канала без памяти может быть сформулирована как

$$\hat{u}_0^{n-1} = \arg \min_{c_0^{n-1} \in \mathcal{C}} E(c_0^{n-1}, S_0^{n-1})$$

Лемма 3. *Для любого* $c_0^{2n-1} \in \mathbb{F}_2^{2n-1}$

$$E(c_0^{2n-1}, S_0^{2n-1}) = E(c_{0,e}^{2n-1} + c_{0,o}^{2n-1}, \tilde{S}_0^{n-1}) + E(c_{0,o}^{n-1}, \overline{S}_0^{n-1}),$$

где $\tilde{S}_i = Q(S_{2i}, S_{2i+1})$, $\overline{S}_i = P(c_{2i}, S_{2i}, S_{2i+1})$.

Доказательство. Достаточно показать справедливость утверждения для $n = 1$. В этом можно убедиться, проверив все возможные комбинации $c_0, c_1 \in \mathbb{F}_2$. □

Полагая $R_m^{(-1)}(y_0^{n-1}) = 0$, и применяя лемму 3 рекурсивно, получим

$$R_m^{(n-1)}(v_0^{n-1}, y_0^{n-1}) = -E(v_0^{n-1} A_m | \mathbf{S}), \tag{4.10}$$

где $\mathbf{S} = (S_0^{(0)}(y_0), \ldots, S_0^{(0)}(y_{n-1}))$. Таким образом, предложенная весовая функция связана с корреляционной невязкой кодового слова $u_0^{n-1} A_m$.

4.1.2.4. Сводка предлагаемого подхода

Предлагается производить декодирование полярных (под)кодов с помощью стекового алгоритма, описанного в разделе 4.1.1, с использованием весовой функции $M_3(v_0^{\phi-1}, y_0^{n-1})$, задаваемой (4.3). Первая компонента весовой функции $M_2(v_0^{\phi-1}, y_0^{n-1}) = R_m^{(\phi-1)}(v_0^{\phi-1}, y_0^{n-1})$ задается (4.5), т.е. она равна накопленному штрафу по фазам $0, \ldots, \phi - 1$. Штраф на некоторой фазе равен (с точностью до знака) абсолютной величине ЛОПП $S_m^{(i)}(v_0^{i-1} | y_0^{n-1})$, если его знак не согласуется с v_i, и нулю в противном случае. В качестве веса пути нулевой длины $R_m^{(-1)}(\epsilon, y_0^{n-1})$ может использоваться любое значение, например 0. Вторая компонента весовой функции равна матожиданию первой компоненты в предположении, что $v_0^{\phi-1} = u_0^{\phi-1}$, т.е. что рассматриваемый путь является префиксом истинного. ЛОПП рекурсивно вычисляются с помощью (1.21)–(1.22).

Отметим, что данный подход может быть использован совместно с классическими полярными кодами, полярными подкодами и полярными кодами с CRC [159]. В случае последних итерации должны выполняться до тех пор, пока не будет получен вектор, удовлетворяющий проверочным соотношениям CRC, или не будет превышено ограничение на максимальное число проходов (L) через каждую фазу.

Для реализации приоритетной очереди в предлагаемом алгоритме декодирования может использоваться подход, описанный в [194].

4.1.3. Блочный последовательный алгоритм

Вычислительная эффективность последовательного алгоритма декодирования может быть повышена за счет использования блоковой обработки символов входного вектора поляризующего преобразования [180]. Рекурсивно применим к рассматриваемому коду обобщенное разложение Плоткина На l-м уровне рекурсии рассматриваемые коды соответствуют некоторым подвекторам (блокам) вектора u_0^{n-1}. Блок с номером $i \in \{0,1\}^l$ включает символы $u_{\phi_i - n_i + 1}, \ldots, u_{\phi_i}$, где ϕ_i — граница блока и n_i — его размер, т.е. длина соответствующего кода $\mathcal{C}_i$. На $(l+1)$-ом уровне рекурсии это блок разбивается на два блока с границами $\phi_{i1} = \phi_i$ и $\phi_{i0} = \phi_i - n_i/2$. При этом на нулевом уровне рекурсии имеется единственный блок с границей $\phi = n - 1$. Каждый такой блок соответствует некоторому коду $\mathcal{C}_i$. Будем останавливать рекурсивное ОРП при получении кодов, допускающих эффективное списочное декодирование. Например, такими кодами являются:

1. код с повторениями;
2. каскадные коды с внешним кодом Рида-Маллера порядка 1 и внутренним кодом с повторениями;
3. коды со скоростью 1;
4. коды с проверкой на четность;
5. расширенный код Хемминга (16,11,4).

Следует отметить, что длины этих кодов могут быть различными. Путем рекурсивного применения леммы 3, несложно получить

Теорема 4. *Пусть i', i — номера двух смежных блоков в дереве рекурсивного ОРП, т.е. $\phi_i = \phi_{i'} + n_i$ и $n_i = 2^{m_i}$. Тогда*

$$R_m^{(\phi_i)}(u_0^{\phi_i} | y_0^{n-1}) = R_m^{(\phi_i')}(u_0^{\phi_{i'}} | y_0^{n-1}) - E(u_{\phi_{i'}+1}^{\phi_i} A_{m_i}, \mathbf{S}),$$

где $\mathbf{S}$ *— вектор значений* $S_{m-m_i}^{(\phi_i/2^{m_i})}$*, задаваемых* (1.21)–(1.22).

Таким образом, декодирование может быть выполнено следующим образом. Пусть $\mathcal{L} = \{\phi_i\}$ — множество границ выявленных блоков. На каждой итерации из приоритетной очереди извлечем путь $v_0^{\phi_i}, \phi_i \in \mathcal{L}$, с наибольшим весом. Для него вычислим вектор ЛОПП $S_{m-m_i}^{(\phi_i/2^{m_i})}$ и подготовимся[3] к декодированию кода $\mathcal{C}_i$. Далее найдем наиболее вероятное кодовое слово $v_{\phi_i+1}^{\phi_i+n_i} A_{m_i}$, соответствующее вычисленным ЛОПП, его корреляционную невязку e_0 и корреляционную невязку e_1 следующего кодового слова. Поместим в ПО пути $v_0^{\phi_i+n_i}$ и $v_0^{\phi_i}$. В качестве веса первого будем использовать значение $R(v_0^{\phi_i}|y_0^{n-1}) - e_0 - \Psi(\phi_i + n_i)$, а в качестве веса второго — $R(v_0^{\phi_i}|y_0^{n-1}) - e_1 - \Psi(\phi_i + n_i)$. Кроме того, сопоставим пути $v_0^{\phi_i}$ переменную состояния декодера $\mathcal{C}_i$. Если при последующих итерациях путь $v_0^{\phi_i}$ будет извлечен из ПО в j-ый раз, $1 \le j \le L$, воспользуемся сохраненным состоянием декодера $\mathcal{C}_i$ для нахождения j-го кодового слова $\widetilde{v}_{\phi_i+1}^{\phi_i+n_i} A_{m_i}$, его корреляционной невязки e_j и корреляционной невязки $(j+1)$-го слова e_{j+1}. Поместим в приоритетную очередь пути $v_0^{\phi_i}.\widetilde{v}_{\phi_i+1}^{\phi_i+n_i}$ с весом $R(v_0^{\phi_i}|y_0^{n-1}) - e_j - \Psi(\phi_i + n_i)$ и $v_0^{\phi_i}$ с весом $R(v_0^{\phi_i}|y_0^{n-1}) - e_{j+1} - \Psi(\phi_i + n_i)$.

Заметим, что данный подход не требует единовременного построения списка кодовых слов внешних кодов, отсортированных в порядке увеличения корреляционных невязок. В большинстве случаев оказывается достаточно построить лишь одно слово (т.е. выполнить декодирование по максимуму правдоподобия). Более того, в условиях слабых шумов может быть достаточно по вектору ЛОПП построить вектор жестких решений и проверить его на принадлежность соответствующему внешнему коду путем умножения на его проверочную матрицу. За счет данных методов можно существенно снизить среднюю сложность декодирования. В худшем случае она, однако, остается равной $O(Ln \log n)$ [180].

4.1.4. Гибридное декодирование

Для эффективной работы блочного последовательного алгоритма декодирования необходимо наличие простых алгоритмов, которые были бы способны породить по мере необходимости (почти) все кодовые слова рассматриваемых кодов $\mathcal{C}_i$ в порядке возрастания их корреляционной невязки.

Рассмотрим линейный $(N = 2^M, k)$ код $\mathcal{C}$ с порождающей матрицей G. Пусть G_0 и G_1 — порождающие матрицы внешних кодов $\mathbb{C}_0, \mathbb{C}_1$, получаемых путем применения обобщенного разложения Плоткина к коду $\mathcal{C}$. Предположим, что имеются алгоритмы декодирования кодов

[3] Содержание подготовки зависит от кода $\mathcal{C}_i$. Она включает в себя все действия, которые необходимы для последующего эффективного нахождения кодовых слов в порядке увеличения их корреляционных невязок.

$\mathbb{C}_0$ и $\mathbb{C}_1$, способные порождать все их кодовые слова в порядке увеличения корреляционной невязки. Такие алгоритмы могут быть получены или путем рекурсивного применения данного метода, или с помощью иных подходов, например, списочного алгоритма Витерби [147]. Предлагаемый подход использует $\mathbb{C}_0$ и l экземпляров декодера $\mathbb{C}_1$, где l — максимальное число кодовых слов, которые требуется построить [91].

Для заданного вектора ЛОПП y_0^{N-1} список наиболее вероятных кодовых слов $\mathcal{C}$ может быть получен путем вычисления $\tilde{y}_i = Q(y_i, y_{i+N/2}), 0 \le i < N/2$, нахождения наиболее вероятных кодовых слов $\tilde{c}^{(j)} = \tilde{u}^{(j)} G_0 \in \mathbb{C}_0, 0 \le j < l$, вычисления

$$\overline{y}_i^{(j)} = (-1)^{(\tilde{u}^{(j)} \tilde{I} G_2)_i} P(\tilde{c}_i^{(j)}, y_i, y_{i+N/2}),$$

и декодирования каждого $\overline{y}^{(j)}$ в коде $\mathbb{C}_1$, что приводит к кодовым словам $\overline{c}^{(j,t_j)} \in \mathbb{C}_1, 0 \le t_j < l$. Наконец, необходимо построить кодовые слова $(\tilde{c}^{(j)} + \overline{c}^{(j,t_j)} + \tilde{u}^{(j)} \tilde{I} G_2, \overline{c}^{(j,t_j)} + \tilde{u}^{(j)} \tilde{I} G_2) \in \mathcal{C}$ и выбрать из них l с наименьшими значениями $E(\tilde{c}^{(j)}, \tilde{y}) + E(\overline{c}^{(j,t_j)}, \overline{y}^{(j)})$. Заметим, что полученный список может не содержать некоторые из l наиболее вероятных кодовых слов $\mathcal{C}$, т.к. они могут соответствовать кодовым словам $\tilde{c}^{(j)} \in \mathbb{C}_0, j \ge l$. Вероятность такого события убывает с увеличением l и растет с увеличением скорости $\mathbb{C}_0$.

Для порождения кодовых слов в порядке увеличения корреляционной невязки описанный алгоритм может быть упрощен. Предположим, что уже построены t_j кодовых слов кода $\mathbb{C}_1$, соответствующих некоторым $\tilde{c}^{(j)} \in \mathbb{C}_0$. Пусть e_j — нижняя граница на корреляционную невязку кодового слова кода $\mathcal{C}$, соответствующего $\tilde{c}^{(j)} \in \mathbb{C}_0$ и $\overline{c}^{(j,t_j)} \in \mathbb{C}_1$. Из леммы 3 следует, что такая граница может быть получена как

$$e_j = \begin{cases} E(\tilde{c}^{(j)}, \tilde{y}), & t_j = 0 \\ E(\tilde{c}^{(j)}, \tilde{y}) + E(\overline{c}^{(j,t_j-1)}, \overline{y}^{(j)}), & t_j > 0. \end{cases}$$

Эта граница может быть использована для отсечения заведомо плохих кодовых слов, как показано в алгоритме, представленном на рис. 4.3. Функция $PreprocessHybrid$ производит инициализацию и предварительную обработку вектора ЛОПП, соответствующего коду $\mathbb{C}_0$. В функции $NextCWHybrid$ находится номер q, минимизирующий $(E(\tilde{c}^{(q)}, \tilde{y}) + E(\overline{c}^{(q,t_q)}, \overline{y}^{(q)}))$, строится соответствующее кодовое слово c и вычисляется его корреляционная невязка w. $Preprocess(\mathbb{C}_i, y)$ обозначает обращение к соответствующей функции для декодера кода $\mathbb{C}_i$, где y — вектор ЛОПП. $NextCW(\mathbb{C}_i, y)$ возвращает пару $(c^{(j)}, u^{(j)}), 0 \le j < l, c^{(j)} \in \mathbb{C}_i$, где $u^{(j)}$ — вектор информационных символов, соответствующих $c^{(j)}$, причем при последовательных обращениях к функции эти пары возвращаются в порядке увеличения $E(c^{(j)}, y)$. Если элемент p_j массива p длины l равен нулю, декодер кода $\mathbb{C}_1$, соответствующий кодовому

PreprocessHybrid$(\mathcal{C}, y_0^{N-1})$

1 $\tilde{y}_i = Q(y_i, y_{i+N/2}), 0 \le i < N/2$
2 Preprocess$(\mathbb{C}_0, \tilde{y})$
3 **for** $j \leftarrow 0$ **to** $l-1$
4 **do** $t_j \leftarrow 0; f_j \leftarrow 0; p_j \leftarrow 0$

NextCWHybrid$(\mathcal{C}, y_0^{N-1})$

1 $w \leftarrow \infty$
2 **for** $j \leftarrow 0$ **to** $l-1$
3 **do if** $f_j = 0$
4 **then** $(\tilde{c}^{(j)}, \tilde{u}^{(j)}) =$ NextCW$(\mathbb{C}_0, \tilde{y})$
5 $E_j = E(\tilde{c}^{(j)}, \tilde{y});\ \hat{c}^{(j)} = \tilde{u}^{(j)}\tilde{I}G_2;\ f_j \leftarrow 1$
6 **if** $f_j = 1$
7 **then if** $E_j \le w$
8 **then for** $i \leftarrow 0$ **to** $\frac{N}{2} - 1$
9 **do** $\overline{y}_i^{(j)} \leftarrow P(\tilde{c}_i^{(j)}, y_i, y_{i+\frac{N}{2}})$
10 $\overline{y}_i^{(j)} \leftarrow (-1)^{\hat{c}_i^{(j)}} \overline{y}_i^{(j)}$
11 Preprocess$(\mathbb{C}_1, \overline{y}^{(j)})$
12 $(\overline{c}^{(j,0)}, \tilde{u}^{(j)}) =$ NextCW$(\mathbb{C}_1, \overline{y}^{(j)})$
13 $e_j \leftarrow E_j + E(\overline{c}^{(j,0)}, \overline{y}^{(j)})$
14 $f_j \leftarrow 2$
15 **else break**
16 **if** $f_j = 2 \wedge p_j = 1 \wedge e_j \le w$
17 **then** $w \leftarrow e_j; q \leftarrow j$
18 $c \leftarrow (\tilde{c}^{(q)} + \overline{c}^{(q,t_q)} + \hat{c}^{(q)}; \overline{c}^{(q,t_q)} + \hat{c}^{(q)})$
19 $t_q \leftarrow t_q + 1$
20 **if** $t_q < l$
21 **then** $\overline{c}^{(q,t_q)} =$ NextCW$(\mathbb{C}_1, \overline{y}^{(q)})$
22 $e_q \leftarrow E_q + E(\overline{c}^{(q,t_q)}, \overline{y}^{(q)})$
23 **else** $p_q \leftarrow 1$
24 **return** c

Рис. 4.3. Гибридный алгоритм декодирования

слову $\tilde{c}^{(j)} \in \mathbb{C}_0$, возвратил менее l кодовых слов. В связи с этим необходимо рассматривать только кодовые слова $\tilde{c}^{(j)}$, соответствующие $p_j = 0$. Если $\tilde{w}^{(k)}$ — минимальная корреляционная невязка среди пар $(\tilde{c}^{(j)}, \overline{c}^{(j,t_j)}), 0 \le j < k$ и $\tilde{w}^{(k)} < E(\tilde{c}^{(k+1)})$, нет необходимости вычислять $\overline{c}^{(k+1,0)}$ и $\tilde{c}^{(k+2)}$. Для того, чтобы реализовать этот подход используется вспомогательный массив f длины l, где

$$f_j = \begin{cases} 0, & \text{если } \tilde{c}^{(j)} \text{ еще не был получен} \\ 1, & \text{если } \tilde{c}^{(j)} \text{ уже получен, но } \overline{c}^{(j,0)} \text{ еще не получен} \\ 2 & \text{, иначе.} \end{cases}$$

Сложность предлагаемого алгоритма декодирования зависит от сложности декодеров кодов $\mathbb{C}_i$. Пусть сложности алгоритмов предварительной обработки и построения кодовых слов равны соответственно $f_i(N,l)$, $b_i(N,l)$. Функция $PreprocessHybrid$ включает вычисление $\tilde{y}$, его предварительную обработку декодером $\mathbb{C}_0$ и инициализацию массивов. Таким образом, ее сложность равна $O(N + f_0(N,l) + l)$. Сложность функции $NextCWHybrid$ есть случайная величина, которая зависит от звшумленного вектора и вероятности ошибки списочного декодирования $\mathbb{C}_0$. В худшем случае должны быть найдены l кодовых слов $\tilde{c}^{(j)} \in \mathbb{C}_0$, вычислены соответствующие векторы ЛОПП $\overline{y}^{(j)}$ и кодовые слова $\overline{c}^{(j,0)} \in \mathbb{C}_1$. Таким образом, сложность составляет $O(l(b_0(N,l) + N + f_0(N,l) + b_1(N,l)))$ операций. Однако в большинстве случаев число обращений к функциям $NextCW$ оказывается намного меньшим [91].

Хотя этот алгоритм может быть использован для декодирования любого кода, его эффективность зависит от параметров кодов, возникающих в обобщенном разложении Плоткина, а именно от распределения числа j, где $\tilde{c}^{(j)} \in \mathbb{C}_0$ — кодовое слово, соответствующее искомому кодовому слову декодируемого кода.

Данный подход был использован при реализации декодеров кодов $(16,10,4), (16,11,4), (16,12,2)$ в вышеописанном блочном последовательном алгоритме декодирования.

4.1.5. Численные результаты

Сложность и корректирующая способность предложенного алгоритма были изучены для случая канала с АБГШ и двоичной фазовой модуляцией. Представлены результаты для полярных кодов с CRC-16 (polar-CRC) и рандомизированных полярных подкодов (ps), описанных в разделе 3.2.1.2. Во всех случаях размер приоритетной очереди был выбран равным $\Theta = Ln$.

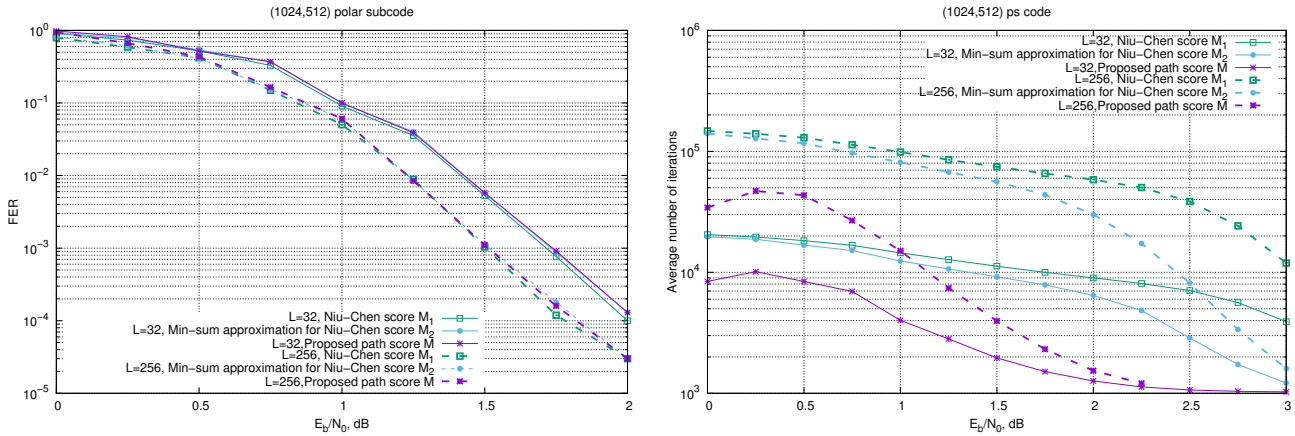


Рис. 4.4. Влияние весовой функции на корректирующую способность и сложность декодера

На рис. 4.4 представлена зависимость вероятности ошибки и среднего числа итераций, выполняемых последовательным алгоритмом декодирования, от отношения сигнал шум для предложенной весовой функции M, алгоритмом Нию-Ченя [131] с весовой функцией

$$M_1(v_0^{\phi-1}, y_0^{n-1}) = \mathcal{W}_m^{(\phi-1)}\left\{v_0^{\phi-1}|y_0^{n-1}\right\},$$

и ее аппроксимацией минимум-сумма

$$M_2(v_0^{\phi-1}, y_0^{n-1}) = R_m^{(\phi-1)}(v_0^{\phi-1}, y_0^{n-1}).$$

Напомним, что предложенная весовая функция может быть представлена как $M_3(v_0^{\phi-1}, y_0^{n-1}) = M_2(v_0^{\phi-1}, y_0^{n-1}) - \Psi(\phi)$. Следует также отметить, что при достаточно большом Θ алгоритм Нию-Ченя обеспечивает ту же корректирующую способность, что и декодер Тала-Варди.

Можно заметить, что использование весовой функции M_2 приводит к незначительному увеличению вероятности ошибки и весьма существенному снижению среднего числа итераций, выполняемых декодером. Это объясняется наличием большого числа путей v_0^{n-1} с малыми вероятностями $\mathcal{W}_m^{(n)}\{v_0^{n-1}|y_0^{n-1}\}$, сумма которых $\mathcal{W}_m^{(\phi)}\left\{v_0^{\phi-1}|y_0^{n-1}\right\} = \sum_{v_\phi^{n-1}} \mathcal{W}_m^{(n)}\{v_0^{n-1}|y_0^{n-1}\}$ может быть весьма значительной. Таким образом, при использовании весовой функции M_1 декодер вынужден обходить множество неверных путей $v_0^{\phi-1}$. На достаточно больших отношениях сигнал/шум вектор v_0^{n-1}, максимизирующий (4.1), с большой вероятностью удовлетворяет всем ограничениям замораживания, поэтому значение функции M_2 оказывается близким к итоговому весу пути. Это позволяет декодеру избежать обхода многих ошибочных путей в кодовом дереве.

Еще более значительное снижение сложности достигается при использовании предложенной весовой функции M. Предложенная весовая функция дает возможность сравнивать

Таблица 4.1. Средняя сложность декодирования $(1024, 512, 28)$ кода с $L = 32$, $\times 10^3$ операций

E_b/N_0, дБ	Сложений		Сравнений	
	Предлагаемая весовая функция	Весовая функция из [119]	Предлагаемая весовая функция	Весовая функция из [119]
0.5	63.2	133	122.5	218
1	34.8	73	55.6	122
1.5	16	32	21.9	54
2	8.8	18	12.0	31

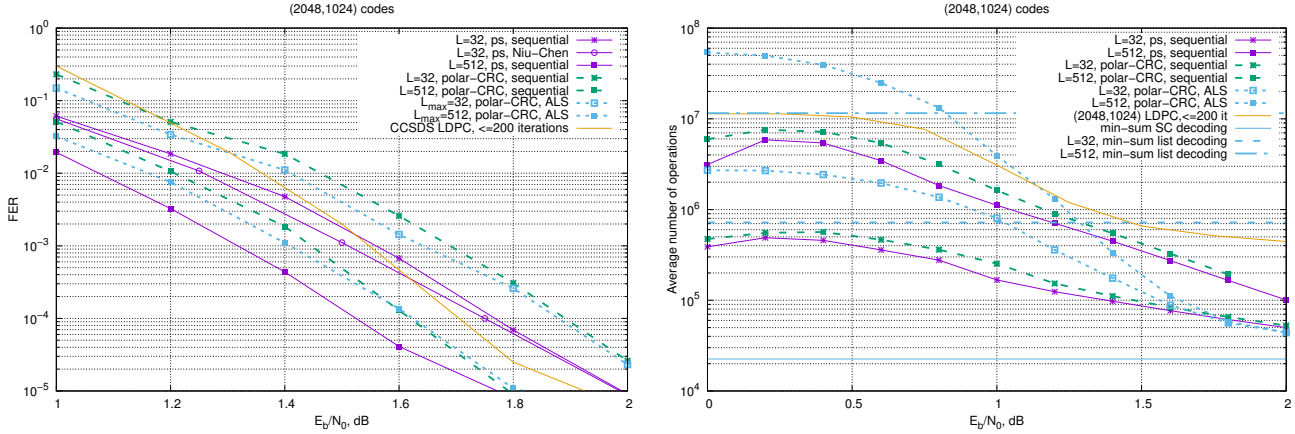


Рис. 4.5. Вероятность ошибки и сложность декодирования кодов $(2048, 1024)$

пути $v_0^{\phi-1}$ различных длин ϕ. В результате достигается многократное снижение среднего числа итераций. Заметим, что корректирующая способность декодера, использующего весовую функцию M, остается примерно такой же, как и в случае весовой функции M_2. В таблице 4.1 представлено сравнение среднего числа арифметических операций, выполняемых последовательным декодером при использовании предложенной весовой функции, а также в случае весовой функции, введенной в [119]. Можно заметить, что использование предложенной весовой функции обеспечивает существенное снижение сложности декодирования.

На рис. 4.5 представлена корректирующая способность и средняя сложность декодирования кодов $(2048, 1024)$. Для сравнения приведены результаты для случая полярных кодов с CRC-16 со списочным декодированием с адаптивным размером списка (ALS) [108], LDPC кода, описанного в стандарте CCSDS, декодирование которого осуществлялось с помощью алгоритма распространения доверия с чередованием [195]. Кроме того, представлена сложность версии “минимум-сумма” алгоритма Тала-Варди с различными размерами списка. Сложность представлена в терминах среднего числа операций сложения и сравнения в слу-

чае полярных (под)кодов и среднего числа операций сложения и вычисления $\log\tanh(x/2)$ в случае LDPC кода.

Можно заметить, что в случае полярного кода с CRC использование последовательного алгоритма декодирования приводит к более заметному увеличению вероятности ошибки по сравнению с алгоритмом Нию-Ченя и списочным декодированием с адаптивным размером списка, чем в случае кода $(1024, 512)$. Это обусловлено отмеченной выше субоптимальностью последовательного декодирования. Это, однако, может быть скомпенсировано за счет использования надлежащим образом построенных полярных подкодов. Следует отметить, что в области высоких отношений сигнал/шум алгоритм ALS имеет несколько меньшую сложность по сравнению с предложенным. Однако этот алгоритм основан на использовании CRC для принятия решения о повторном декодировании с большим размером списка, и его обобщение на случай полярных подкодов не является очевидным. В области малых отношения сигнал/шум, где вероятность ошибки превышает 10^{-3}, предложенный алгоритм имеент меньшую среднюю сложность по сравнению с ALS и в случае полярных подкодов обеспечивает энергетический выигрыш 0.1 дБ..

Следует также отметить, что среднее число сложений и умножений в случае предложенного алгоритма быстро сходится к сложности версии “минимум-сумма” декодера ПИ. Можно также заметить, что полярные подкоды в сочетании с предложенным алгоритмом при $L = 32$ обеспечивают корректирующую способность с одним из лучших известных LDPC кодов, а при больших L обеспечивают значительный энергетический выигрыш. При этом средняя сложность предложенного алгоритма декодирования оказывается значительно меньше средней сложности декодирования LDPC кодов.

На рис. 4.6 представлено распределение числа итераций, выполняемых декодером в случае правильного и ошибочного декодирования. Можно заметить, что распределение имеет достаточно “тяжелый хвост”. В случае ошибки декодирования число итераций может оказаться весьма большим. Вместе с тем, для рассмотренного случая вероятность того, что декодирование будет правильным при условии того, что декодер выполнил более 20000 итераций, является пренебрежимо малой. Это можно использовать для раннего останова с отказом от декодирования.

Рис. 4.7, *а* иллюстрирует корректирующую способность и сложность блочного последовательного алгоритма декодирования (BSDA) с параметрами $L = 32, \Theta = 1024$. Для сравнения приведены также результаты для быстрого списочного алгоритма для полярных кодов с CRC и $L = 2$, заимствованные из [62]. Как и следовало ожидать, большее значение L позволяет предложенному алгоритму обеспечивать существенно лучшую корректирующую способность

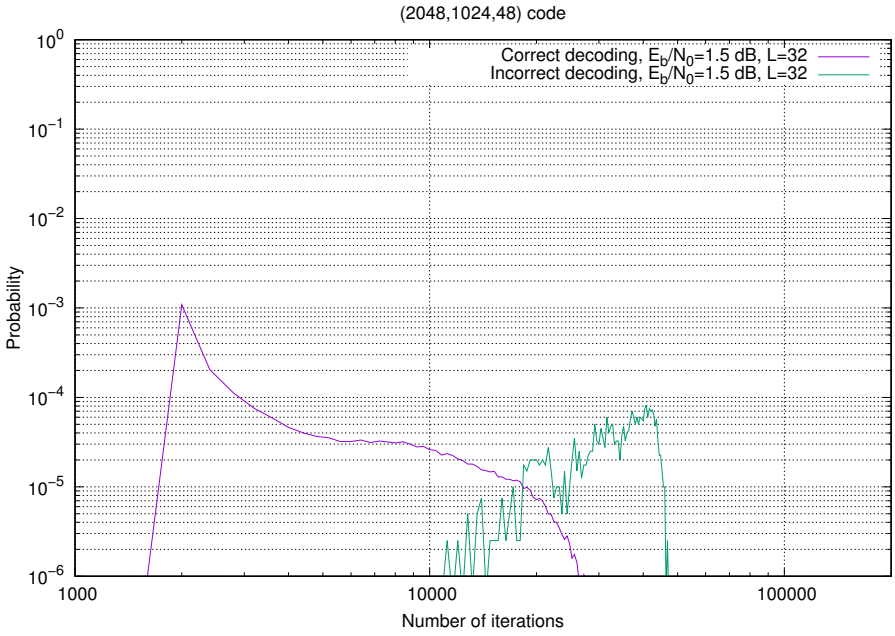
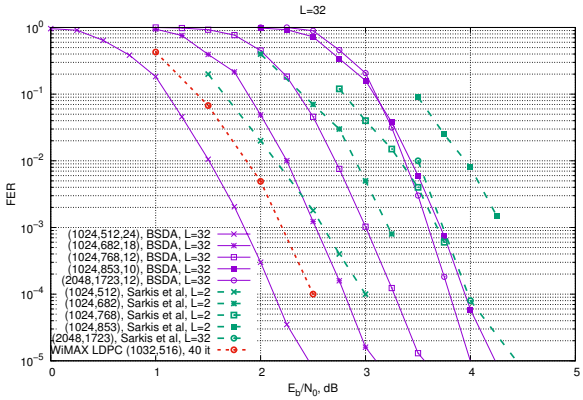


Рис. 4.6. Распределение числа итераций при последовательном декодировании

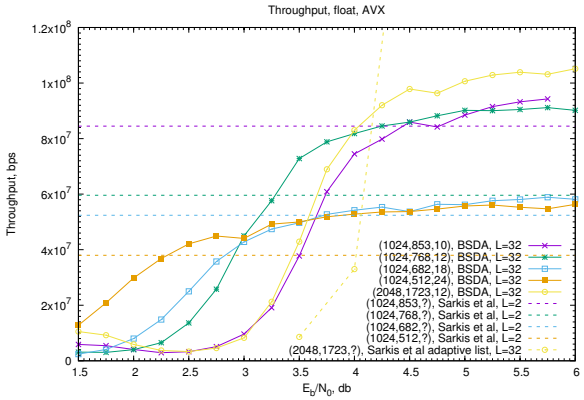
по сравнению с быстрым списочным алгоритмом.

На рис. 4.7, *б* приведены результаты, характеризующие производительность программной реализации предложенного алгоритма. Исследования проводились на ПК, оснащенном процессором Intel Core i7-2600K с тактовой частотой 3.4 ГГц. Для сравнения представлены также результаты из [62], полученные на ПК того же типа. Можно заметить, что при достаточно большом отношении сигнал/шум блочный последовательный алгоритм с $L = 32$ обеспечивает существенно большую производительность по сравнению с быстрым списочным декодером из [62] с $L = 2$. Для кода $(2048, 1723)$ на высоких ОСШ адаптивный списочный декодер с $L = 32$ обеспечивает лучшую производительность, чем блочный последовательный алгоритм. Следует отметить, что подход, предложенный в [62] основан на использовании "раскрученного" декодера последовательного исключения, не включающего ни одного условного оператора, автоматически сгенерированного для конкретного кода. Только если данный алгоритм оказывается неспособен найти кодовое слово с правильным значением CRC, задействуется списочный декодер. Предлагаемый алгоритм является универсальным и при этом оказывается более эффективным в области умеренных ОСШ.

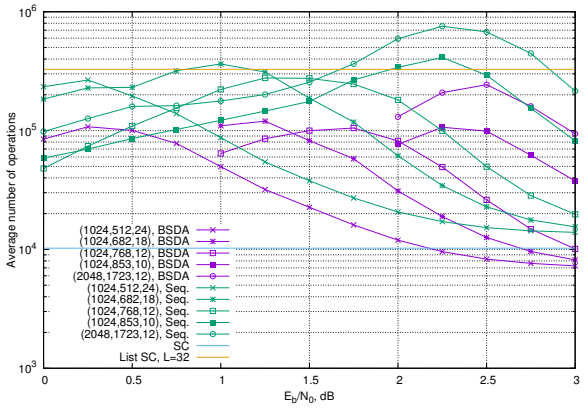
На рис. 4.7, *в* показано среднее число сложений и сравнений, выполняемое предложенным алгоритмом. Для сравнения приведены также результаты для последовательного алгоритма декодирования. Можно заметить, что с увеличением ОСШ сложность сходится к величине несколько меньшей, чем $n \log_2 n$ (сложность алгоритма ПИ). Это снижение сложности достигается за счет использования специализированных алгоритмов декодирования



(*а*) Вероятность ошибки



(*б*) Производительность



(*в*) Среднее число операций

Рис. 4.7. Корректирующая способность и сложность блочного последовательного алгоритма декодирования

внешних кодов, возникающих в ОРП рассматриваемых полярных подкодов.

4.2. Декодирование цепных полярных подкодов

Рассмотрим декодирование цепных полярных подкодов, введенных в разделе 3.2.2.

4.2.1. Обобщенный алгоритм последовательного исключения

Пусть W — матрица прекодирования цепного полярного подкода, и пусть l_j — номер столбца, содержащего первый ненулевой элемент строки j этой матрицы. Множество замороженных символов цепного полярного подкода может быть определено как

$$\mathcal{F} = \{0, \ldots, n-1\} \setminus \{l_j | 0 \le j < k\}\,.$$

Не ограничивая общности, будем считать, что столбцы l_j матрицы W имеют вес 1.

Определение 10. *Расписанием декодера* Φ *для* $k \times n$ *матрицы прекодирования* W *называется последовательность различных пар* $\Phi_s = (i_s, \phi_s), 0 \le s < n, 0 \le i_s < z, 0 \le \phi_s < n_{i_s}$, *удовлетворяющая следующим условиям:*

1. *если* $s' < s''$ *и* $i_{s'} = i_{s''}$, *то* $\phi_{s'} < \phi_{s''}$,
2. *для любых* s', s'' *таких, что* $\Psi(\Phi_{s'}) = l_j$ *для некоторого* $j : 0 \le j < k$, *и* $t = \Psi(\Phi_{s''}) > \Psi(\Phi_{s'})$, *где* $W_{j,t} = 1$, *выполняется* $s' < s''$.

Здесь под

$$\Psi(\Phi_s) = \Psi(i_s, \phi_s) = \phi_s + \sum_{j=0}^{i_s-1} 2^{m_j}$$

понимается глобальная фазовая функция.

Будем называть естественным расписанием декодера последовательность $[(0,0), \dots, (0, n_0 - 1), \dots, (z-1, 0), \dots, (z-1, n_{z-1} - 1)]$. Для такого расписания выполняется $\Psi(\Phi_s) = s$.

Введем обозначения $\tilde{n}_s = \sum_{j=0}^{i_s-1} 2^{m_j}$ и $\overline{n}_s = \sum_{j=0}^{i_s} 2^{m_j}$.

Для некоторого фиксированного расписания Φ декодирование цепного полярного подкода может быть выполнено с помощью следующего обобщенного алгоритма последовательного исключения:

1. Пусть $\hat{u}_i \leftarrow 0, i \in \mathcal{F}$
2. Для s с 0 по $n-1$, удовлетворяющих $\Psi(\Phi_s) \notin \mathcal{F}$:
   - Присвоить
   $$\hat{u}_i \leftarrow \arg\max_{u_i \in \mathbb{F}_2} \mathbf{W}_{m_{i_s}}^{(\phi_s)}(y_{\tilde{n}_s}^{\overline{n}_s-1}, \hat{u}_{\tilde{n}_s}^{i-1} | u_i),$$
   где $i = \Psi(\Phi_s)$.
   - Присвоить $\hat{u}_j \leftarrow \hat{u}_j + \hat{u}_i$ для всех таких $j > i$, что $W_{t_i,j} = 1$, где t_i — номер строки матрица W, содержащей первый ненулевой элемент в столбце i.

Несложно переписать этот алгоритм в виде, аналогичном (2.4).

Вероятность ошибки данного алгоритма может быть оценена как

$$P_{SC} \le \sum_{j=0}^{k-1} \Pi_{l_j},$$

где

$$\Pi = (P_{m_0,0}, \dots, P_{m_0,n_0-1}, \dots, P_{m_{z-1},0}, \dots, P_{m_{z-1},n_{z-1}-1}).$$

Видно, что P_{SC} не зависит от расписания декодера. Она минимизируется выбором в качестве W матрицы со строками веса 1, имеющими единицы в столбцах j с наименьшими значениями Π_j. Это соответствует случаю независимого кодирования данных z классическими полярными кодами Арикана длины $2^{m_i}, 0 \le i < z$. Предложенные выше цепные полярные подкоды не могут обеспечить меньшую вероятность ошибки декодирования *методом ПИ*, чем чередование z независимых классических полярных кодов длин 2^{m_j}. Однако значительный выигрыш может быть получен при использовании списочного декодирования.

4.2.2. Списочное декодирование

В данном разделе представлено обобщение списочного алгоритма Тала-Варди на случай цепных полярных подкодов. Для простоты ограничимся рассмотрением его аппроксимации минимум-сумма [35]. Определение метрики пути (1.20) может быть непосредственно обобщено на случай списочного декодирования цепных полярных подкодов с любым расписанием декодера.

Предлагаемый декодер состоит из z экземпляров списочного декодера Тала-Варди (ЭСДТВ) для поляризующих преобразований $A_{m_i}, 0 \le i < z$, которые осуществляют расчет логарифмических отношений правдоподобия для различных путей. При этом операции клонирования и удаления путей производятся синхронно во всех ЭСДТВ. Пусть E_i — ЭСДТВ для поляризующего преобразования A_{m_i}. Под $E_i.FunctionName$ будем понимать функцию, которая работает с данными, относящимися к этому преобразованию.

На рис. 4.8 представлено предлагаемое обобщение алгоритма Тала-Варди. Входными параметрами являются зашумленный вектор y_0^{n-1} и размер списка L. Заметим, что функция $AssignInitialPath$ на каждой итерации возвращает одно и то же значение Λ. Можно упростить реализацию, удалив одинаковые операции, выполняемые для всех E_i. Это, однако, потребует множества мелких изменений в различных вспомогательных алгоритмах. Для простоты изложения будем по возможности следовать описанию исходного алгоритма Тала-Варди.

Функция $RecursivelyCalcS$ вычисляет ЛОПП $S_{m_{i_s}}^{(\phi_s)}$ согласно (1.21)–(1.22). Заметим, что порядок вызова этой функции (см. строки 6 и 16) отличается от такового в алгоритме Тала-Варди. Это необходимо для обеспечения согласованности в структурах данных, обеспечивающих разделение памяти между различными путями.

Функции $ContinuePaths_FrozenBit$ и $ContinuePaths_UnfrozenBit$ расширяют все активные пути $u_{\tilde{n}_s}^{\tilde{n}_s+\phi_s-1}$ на 1 бит и вычисляют обновленные веса путей $R_\Lambda, 0 \le \Lambda < L$, согласс-

```
DECODE(y_0^{n-1}, L)
 1  ν ← 0
 2  for i ← 0 to z − 1
 3  do Λ ← E_i.ASSIGNINITIALPATH()
 4       S = E_i.GETARRAYPOINTERS(0, Λ)
 5       S[j] = y_{ν+j}, 0 ≤ j < 2^{m_i}
 6       E_i.RECURSIVELYCALCS(m_i, 0)
 7       ν ← ν + 2^{m_i}
 8  ActivePath[Λ] = true; R_Λ = 0
 9  for s ← 0 to n − 1
10  do if Ψ(i_s, φ_s) ∈ F
11          then E_{i_s}.CONTINUEPATHS_FROZENBIT(φ_s)
12          else E_{i_s}.CONTINUEPATHS_UNFROZENBIT(φ_s)
13       if φ_s ≡ 1 mod 2
14          then E_{i_s}.RECURSIVELYUPDATEC(m_{i_s}, φ_s)
15       if φ_s < 2^{m_s} − 1
16          then E_{i_s}.RECURSIVELYCALCS(m_{i_s}, φ_s + 1)
17  Λ_0 = arg max_s R_s
18  for i ← 0 to z − 1
19  do C^{(i)} = E_i.GETARRAYPOINTERC(0, Λ_0)
20  return [C^{(0)}, ..., C^{(z−1)}]
```

Рис. 4.8. Обобщенный списочный алгоритм Тала-Варди декодирования цепных полярных подкодов

но (4.3). Функция $ContinuePaths_FrozenBit$ осуществляет расчет значений ДЗС согласно (2.3), в то время как $ContinuePaths_UnfrozenBit$ рассматривает два возможных значения $u_{\Psi(\Phi_s)}$ для каждого пути и выбирает L путей с наибольшими значениями веса R_Λ. После обработки всех фаз на строках 17–20 осуществляется выбор наиболее вероятного кодового слова.

Сложность обобщенного алгоритма Тала-Варди равна $O(L\sum_{i=0}^{z-1} n_i \log n_i)$. Она может быть сокращена за счет использования вышеописанных методов последовательного декодирования.

4.2.3. Жадное расписание

Эксперименты показывают, что корректирующая способность списочного алгоритма декодирования цепных полярных подкодов существенно зависит от используемого расписания декодера. Вероятность потери декодером правильного пути определяется тем, насколько быстро в ходе работы декодера веса неправильных путей становятся достаточно малыми. С

другой стороны, оценки $\hat{u}_{\Psi(\Phi_s)}$, формируемые вышеописанным обобщенным алгоритмом ПИ, не являются оптимальными, т.к. не учитывают ограничения, которые могут быть наложены на некоторые символы $u_{\Psi(\Phi_{s'})}, s' > s$. В связи с этим возникает естественная конструкция жадного расписания декодера. Оно строится таким образом, чтобы замороженные символы обрабатывались как можно раньше, с учетом ограничений, приведенных в определении 10.

Для построения такого расписания предлагается следующий алгоритм:

1. Присвоить $w_i \leftarrow \mathrm{wt}(W_{-,i}), 0 \le i < n$.
2. Присвоить $w_i \leftarrow w_i - 1, i \notin \mathcal{F}$
3. Пусть r_i — мощность наибольшего множества $T_i = \{i+1, i+2, \ldots, j\}$, удовлетворяющего $T_i \cap \mathcal{F} = \varnothing$.
4. Присвоить $\phi_j \leftarrow 0, 0 \le j < z$.
5. Присвоить $s \leftarrow 0$.
6. Присвоить $\Phi_s \leftarrow (j, \phi_j)$, где j является таким целым числом, что существует такое $i = \Psi(j, \phi_j)$, что $w_i = 0$, и r_i является наименьшим возможным.
7. Присвоить $\phi_j \leftarrow \phi_j + 1$.
8. Если $i \notin \mathcal{F}$, присвоить $w_p \leftarrow w_p - 1$ для всех $p : W_{t_i,p} = 1$, где t_i — номер строки матрицы W, первый ненулевой элемент которой расположен в столбце i.
9. Присвоить $s \leftarrow s + 1$. Если $s < n$, перейти к шагу 6.

Здесь w_i равно числу необработанных незамороженных символов, от которых зависит u_i, r_i — число незамороженных символов, которые непосредственно следуют за u_i, т.е. расстояние до ближайшего замороженного символа $u_{j+1}, j \ge i$.

Не представляется возможным ни доказать оптимальность этого алгоритма, ни оценить теоретически корректирующую способность списочного декодера, использующего построенное таким образом расписание. Однако, как показывают результаты статистического моделирования, использование предложенного подхода дает достаточно хорошие результаты.

4.2.4. Численные результаты

На рис. 4.9 представлены результаты статистического моделирования для случая канала с АБГШ и двоичной фазовой модуляцией. Рассмотрены случаи естественного и жадного

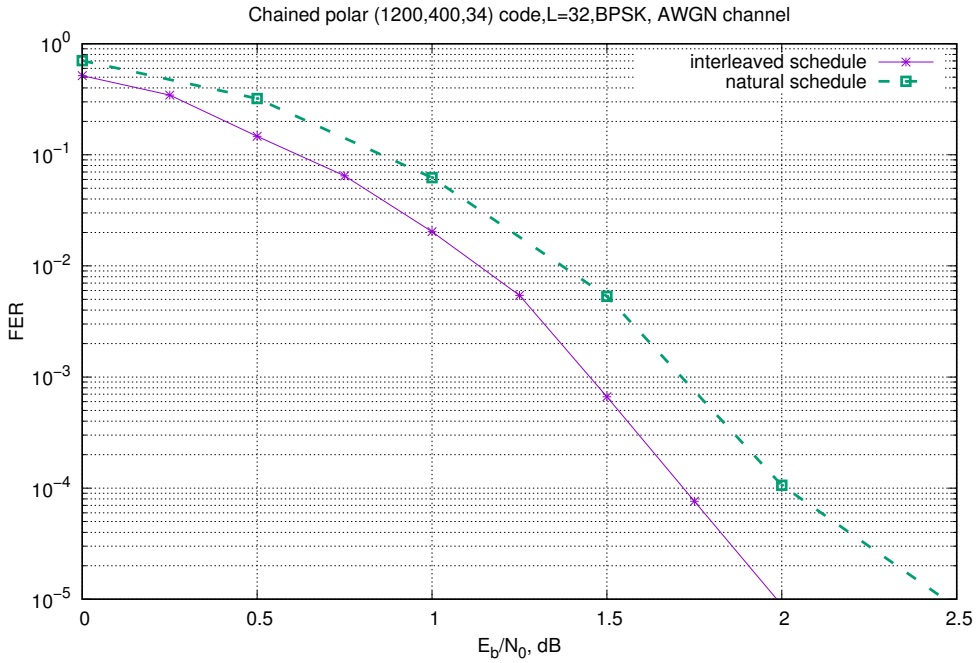


Рис. 4.9. Влияние расписания декодера на вероятность ошибки

расписаний декодера. Видно, что использование жадного расписания позволяет получить энергетический выигрыш до 0.25 дБ. Результаты моделирования, приведенные в разделе 3.2.2, также получены с использованием жадного расписания.

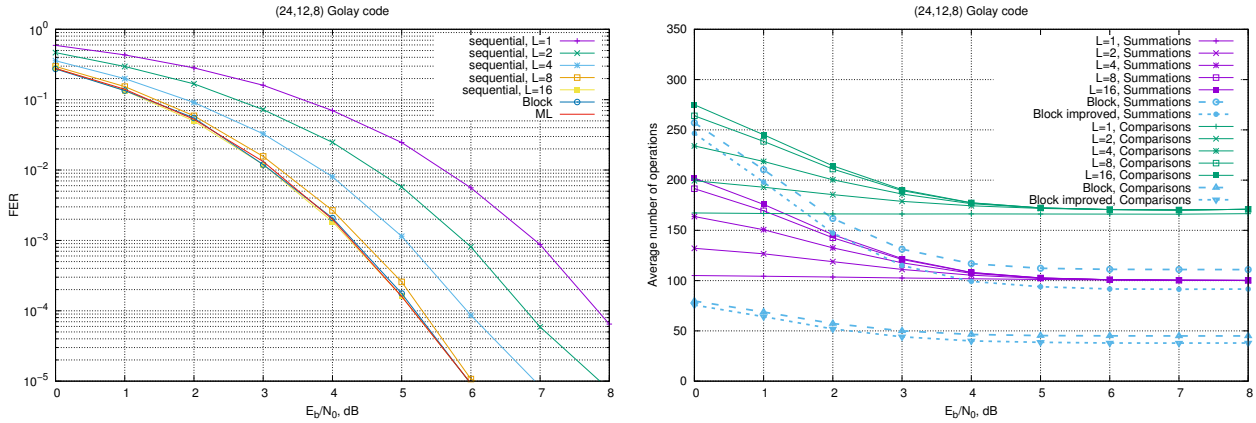
4.3. Декодирование расширенного кода Голея

Рассмотрим декодирование расширенного кода Голея с помощью представления, полученного в п. 2.5.

4.3.1. Списочное и последовательное декодирование

Воспользовавшись методом построения расписания, описанным в разделе 4.2, можно получить следующую последовательность обхода входных символов смешанного поляризующего преобразования (т.е. расписание): $0, 1, 2, 16, 3, 17, 4, 5, 18, 6, 7, 8, 9, 19, 20, 10, 21, 11, 12, 22, 13, 14, 15, 23$. Например, четыре начальных шага списочного декодера в соответствии с этим расписанием, соответствуют замороженным символам $u_0 = u_1 = u_2 = u_{16} = 0$. Таким образом, получается единственный нулевой путь с весом (в случае декодера минимум-сумма, описанного в п. 1.4.4.2)

$$\mathcal{A} = R(0000|y_0^{23}) = \tau(S_4^{(0)}(y_0^{15}), 0) + \tau(S_4^{(1)}(0, y_0^{15}), 0) + \tau(S_4^{(2)}(00, y_0^{15}), 0) + \tau(S_3^{(0)}(y_{16}^{23}), 0).$$



($а$) Вероятность ошибки ($б$) Сложность декодирования

Рис. 4.10. Корректирующая способность последовательного декодера кода Голея в канале с АБГШ

Далее необходимо рассмотреть два возможных значения u_3, т.е. клонировать путь. Это дает возможность обработать условие $u_{17} = u_3$, вытекающее из $u_0^{23}V^T = 0$. Таким образом, получим

$$R(0000u_3u_{17}|y_0^{23}) = \mathcal{A} + \tau(S_4^{(3)}(000, y_0^{15}), u_3) + \tau(S_3^{(1)}(y_{16}^{23}), u_{17}).$$

Для снижения сложности может быть использован последовательный алгоритм декодирования, описанный в разделе 4.1. На рис. 4.10 представлены результаты моделирования для этого случая. Для сравнения на рис. 4.10 приведена также кривая вероятности ошибки декодирования данного кода по максимуму правдоподобия. Видно, что последовательный алгоритм с размером списка $L = 16$ обеспечивает декодирование по максимуму правдоподобия.

4.3.2. Блочное декодирование

Возможен, однако, и более эффективный подход. Анализируя структуру матрицы ограничений V, можно заметить, что выкалывание последних 8 символов приводит к $(16, 11, 4)$ коду Рида-Маллера второго порядка. Для порождения кодовых слов этого кода в порядке увеличения их корреляционной невязки с принятой последовательностью может быть использован алгоритм, описанный в разделе 4.1.4.

Заметим также, что выкалывание всех символов, кроме тех, которые имеют номера $16, \ldots, 23$, приводит к коду $(8, 7, 2)$, который может быть получен с помощью конструкции Плоткина из $(4, 3, 2)$ кода Рида-Маллера 1 порядка и кода $(4, 4, 1)$. Строки 6,7 матрицы V задают линейные соотношения между кодовыми словами кодов $(8, 4, 4)$ и $(4, 3, 2)$.

Корреляционные метрики для кодовых слов кода Рида-Маллера 1 порядка $\mathcal{C}$ длины N

$$\mathbf{C}(c^{(i)}, z_0^{N-1}) = \sum_{j=0}^{N} (-1)^{c_j^{(i)}} z_i, c^{(i)} \in \mathcal{C},$$

где z_i — ЛОПП, могут быть вычислены с помощью быстрого преобразования Адамара порядкаN со сложностью $N \log_2 N$ сложений. Соответствующие корреляционные невязки равны

$$E(c^{(i)}, z_0^{N-1}) = \frac{1}{2}\left(\sum_{j=0}^{N-1} |z_j| - \mathbf{C}(c^{(i)}, z_0^{N-1})\right).$$

Отсюда следует, что

$$2R(u_0^7, u_{16}^{19}|y_0^{23}) = -\sum_{j=0}^{11} |z_j| + C(u_0^7 A_3, z_0^7) + \mathbf{C}(u_{16}^{19} A_2, z_8^{11}), \tag{4.11}$$

где $z_i = S_1^{(0)}(y_{2i}, y_{2i+1}) = S_0^{(0)}(y_{2i}) \boxplus S_0^{(0)}(y_{2i+1}), 0 \le i < 12$, и $u_0 = u_1 = u_2 = u_4 = u_{16} = 0$, $u_3 = u_{17}$, $u_5 = u_{18}$. Заметим, что первое слагаемое не зависит от u_0^{23}, и может быть положено равным 0. С учетом этого, получим, что $R(u_0^6, u_7 = 1, u_{16}^{18}, u_{19} = 1|y_0^{23}) = -R(u_0^6, u_7 = 0, u_{16}^{18}, u_{19} = 0|y_0^{23})$. Таким образом, веса 32 путей (u_0^7, u_{16}^{19}) могут быть найдены с помощью БПА порядка 8 и 4, а также 16 дополнительных сложений. Эти пути могут быть отсортированы[4] в порядке убывания их весов, после чего к ним может быть последовательно применен нижеописанный второй этап обработки.

Для любого пути (u_0^7, u_{16}^{19}) с весом $r = 2R(u_0^7, u_{16}^{19}|y_0^{23})$ можно вычислить $u_9 = u_{20} = u_3 + u_5 + u_6 + u_{19}$. Вычислим далее $\widetilde{z}_{8+i} = S_1^{(1)}(u_{16}^{19} A_3, y_{16+2i}^{16+2i+1}), 0 \le i < 3$. Эти значения могут рассматриваться как ЛОПП для кодового слова смежного класса, задаваемого значением u_{20}, кода $(4, 3, 2)$. Следовательно, соответствующие корреляционные метрики могут быть найдены с помощью БПА порядка 4, после чего веса путей получаются как

$$\rho = 2R(u_0^7, u_{16}^{23}|y_0^{23}) = r - \sum_{i=8}^{11} |\widetilde{z}_i| + \mathbf{C}(u_{20}^{23} A_3, \widetilde{z}_8^{11}).$$

Заметим, что достаточно ограничиться рассмотрением векторов с $\mathbf{C}(u_{20}^{23} A_3, \widetilde{z}_8^{11}) \ge 0$, т.к. u_{23} не заморожен Пусть векторы u_{20}^{23} упорядочены по убыванию $\mathbf{C}(u_{20}^{23} A_3, \widetilde{z}_8^{11})$. Вычислим $u_{10} = u_3 + u_5 + u_{21}$ и $u_5 = u_{22}$, а также $\widetilde{z}_i = S_1^{(1)}(u_0^7 A_4, y_{2i}^{2i+1}), 0 \le i < 7$. Эти значения могут рассматриваться как ЛОПП для кодового слова смежного класса, задаваемого $u_9.u_{10}, u_{12}$, $(8, 4, 4)$ кода Рида-Маллера 1 порядка. Следовательно, можно воспользоваться БПА порядка 8 для расчета соответствующих корреляционных метрик и окончательно выбрать кодовое слово с наибольшим значением

$$2R(u_0^{23}|y_0^{23}) = r - \sum_{i=0}^{11} |\widetilde{z}_i| + \mathbf{C}(u_{20}^{23} A_3, \widetilde{z}_8^{11}) + \mathbf{C}(u_8^{15} A_4, \widetilde{z}_0^7).$$

[4] Заметим, что достаточно отсортировать лишь 16 значений.

Заметим, что коэффициенты 2 и 1/2 в вышеприведенных выражениях могут быть исключены.

Для избежания избыточных вычислений можно хранить наибольшее значение R_{max} величин $R(u_0^{23}|y_0^{23})$, полученных за время работы алгоритма, и прекратить обработку векторов u_{20}^{23} сразу после получения $\rho < R_{max}$, а векторов (u_0^7, u_{16}^{19}) — сразу после получения $r < R_{max}$.

Наименьшая сложность в описанном алгоритме достигается в тех случаях, когда правильное кодовое слово соответствует наибольшим значениям $\mathbf{C}(u_0^7 A_3, z_0^7) + \mathbf{C}(u_{16}^{19} A_2, z_8^{11})$ и $\mathbf{C}(u_{20}^{23} A_3, \widetilde{z}_8^{11})$. В этом случае достаточно вычислить два БПА порядка 4 и 3. Всего при этом алгоритм требует 111 сложений и 45 сравнений.

На высоких отношениях сигнал/шум сложность декодирования может быть сокращена путем построения вектора жестких решений для $\widetilde{z}_0^{11}$, соответствующего некоторому пути (u_0^7, u_{16}^{19}), и вычисления соответствующих значений u_{20}^{22}. Если полученный вектор удовлетворяет ограничениям, задаваемым матрицей V, вычисление БПА на втором этапе алгоритма может быть опущено.

На рис. 4.10, *б* представлено среднее число арифметических операций, выполняемых предложенными алгоритмами декодирования. Можно заметить, что сложность быстро убывает с увеличением отношения сигнал/шум. На высоких ОСШ она приближается к сложности алгоритма Варди, являющегося наиболее эффективным методом декодирования расширенного кода Голея [185], который требует 121 операции. Улучшенный блочный алгоритм декодирования, использующий жесткие решения (HDD) для избежания вычисления БПА на втором этапе, обеспечивает снижение сложности примерно на 20%.

Максимальная сложность блочного алгоритма, наблюдавшаяся при моделировании, составляет 1590 операций, что близко к сложности основанного на БПА алгоритма, описанного в [40].

4.4. Параллельное списочное декодирование звездных полярных подкодов

Пусть $Y = (y^{(0)}, \ldots, y^{(s-1)})$ — зашумленный вектор, полученный в результате передачи кодового слова (3.26) симметричного звездного полярного подкода по каналу без памяти. Целью декодера является нахождение $(u^{(0)}, \ldots, u^{(s-1)})$ с наивысшим значениям

$$R(u^{(0)}, \ldots, u^{(s-1)}|Y) = \sum_{i=0}^{s-1} \log \mathcal{W}_{m_i}^{(\nu_i - 1)}(u^{(i)}|y^{(i)}),$$

при условии выполнения (3.27)–(3.28).

В данном разделе будет описана параллельная версия списочного алгоритма Тала-Варди для случая симметричных звездных полярных подкодов. Для простоты изложения положим $m_i = m, 0 \le i < s$, но описываемый метод может быть применен и в случае лучей различной длины. Основная идея нижеизложенного алгоритма состоит в построении для каждой фазы ϕ списков различных частичных векторов $u^{(i)}$ (т.е. первых их ϕ элементов), и расчете для каждого такого вектора верхней границы для $\mathcal{W}_{m_i}^{(\nu_i-1)}(u^{(i)}|y^{(i)})$. Уравнения перекрестных проверок (3.30) используются на каждой фазе для выбора только таких частичных векторов, которые могут соответствовать допустимому кодовому слову рассматриваемого кода.

Предлагаемый алгоритм декодирования использует s экземпляров списочного декодера Тала-Варди (ЭСДТВ), которые производят расчет логарифмических отношений правдоподобия и управление памятью. i-ый экземпляр принимает $y^{(i)}$ и строит не более L кандидатов для $u^{(i)}$, удовлетворяющих (3.27). Пусть $U_{\phi,j}^{(i)} \in \mathbb{F}_2^{\phi+1}, 0 \le j < L$, — пути, рассматриваемые i-ым ЭСДТВ на фазе ϕ.

На фазе ϕ каждый ЭСДТВ вычисляет *локальный вес* $M_2(U_{\phi,j}^{(i)}, y^{(i)}) = R_{m_i}^{(\phi)}(U_{\phi,j}^{(i)}, y_0^{n-1})$, задаваемый (4.5), для каждого из рассматриваемых путей $U_{\phi,j}^{(i)}$. Напомним, что $M_2(U_{\phi,j}^{(i)}, y^{(i)})$ является верхней границей для $\log \mathcal{W}_{m_i}^{(\nu_i-1)}(u^{(i)}|y^{(i)})$ по всем $u^{(i)}$, первые $\phi+1$ элементов которых совпадают с $U_{\phi,j}^{(i)}$.

Если $\phi \in \{\tau_{i,j}\}$ для некоторого i, то каждый путь $U_{\phi,j}^{(i)}$ дополняется значением соответствующего замороженного символа, задаваемым (3.29). В противном случае, пути $U_{\phi,j}^{(i)}$ клонируются и производится синхронизация ЭСДТВ. Синхронизация состоит в выборе тех путей, которые могут быть скомбинированы в допустимое кодовое слово симметричного звездного полярного подкода, т.е. удовлетворяют (3.29)–(3.30).

Пусть $g(i,\phi) = \{t|\sigma_{i,0,t} \le \phi \wedge \sigma_{0,i,t} \le \phi\}$ — множество уравнений (3.30), которые могут быть проверены на фазе ϕ. Пусть

$$z_{i,\phi}(U_{\phi,j}^{(i)}) = U_{\phi,j}^{(i)}(S^{(i)})_{g(i,\phi),\phi}^T$$

частичный вектор перекрестных проверок пути $U_{\phi,j}^{(i)}$, где $(Q)_{B,\phi}$ — подматрица Q, содержащая строки из множества B и столбцы $0,\dots,\phi$. Определим *комбинированный вес*

$$\beta_{i,\phi}(x|y^{(i)}) = \max_{\substack{0\le j<L \\ z_{i,\phi}(U_{\phi,j}^{(i)})=x}} R(U_{\phi,j}^{(i)}|y^{(i)}), x \in \mathbb{F}_2^{|g(i,\phi)|}.$$

Здесь максимизация производится по множеству путей, построенных i-ым ЭСДТВ на фазе ϕ, имеющим некоторое фиксированное значение частичного вектора перекрестных проверок. Можно заметить, что $\beta_{i,j}(x|y^{(i)})$ является верхней границей логарифмической вероятности путей, имеющих частичный вектор перекрестных проверок x. Если не существует пути в

i-ом ЭСДТВ с $z_{i,\phi}(U_{\phi,j}^{(i)}) = x$, будем считать, что $\beta_{i,\phi}(x|y^{(i)}) = -\infty$.

Если $|g(i,\phi)|$ невелика, то все значения $\beta_{i,\phi}(x|y^{(i)})$ могут быть найдены из $R(U_{\phi,j}^{(i)}|y^{(i)})$ со сложностью $O(L)$ с помощью алгоритма, аналогичного сортировке методом подсчета. В противном случае должен использоваться универсальный алгоритм сортировки со сложностью $O(L\log L)$ или техники хеширования.

Глобальным весом пути $U_{\phi,j}^{(i)}$ будем называть верхнюю границу логарифма вероятности кодового слова, которое может быть получено путем комбинирования $U_{\phi,j}^{(i)}$ с путями $U_{\phi,j'}^{(i')}$, найденными всеми остальными ЭСДТВ $i' \neq i$, т.е.

$$G_i^{(\phi)}(U_{\phi,j}^{(i)}|Y) = R(U_{\phi,j}^{(i)}|y^{(i)}) + \sum_{i'\neq i} \beta_{i',\phi}\left(z_{i,\phi}\left(U_{\phi,j}^{(i)}\right)|y^{(i')}\right). \tag{4.12}$$

Предлагается использовать $G_i^{(\phi+1)}(u_0^{\phi}|Y)$ вместо $R(U_{\phi,j}^{(i)}|y^{(i)})$ в каждом из ЭСДТВ для сортировки и удаления путей. В остальном детали реализации предложенного алгоритма совпадают с [159].

После того, как все ЭСДТВ обработали последнюю фазу $\nu-1$, кодовое слово (3.26) симметричного кода с наибольшим $R(u^{(0)},\ldots,u^{(s-1)}|Y)$ формируется из элементов списков $U_{\phi,j}^{(i)}$, с учетом ограничений (3.28). Данная задача может быть решена путем поиска наилучшего значения вектора перекрестных проверок

$$\hat{x} = \arg\max_{x} \sum_{i=0}^{s-1} \beta_{i,\nu-1}(x|y^{(i)}),$$

и последующего выбора

$$\hat{u}^{(i)} = \arg \max_{j: z_{i,\nu-1}(U_{\phi,j}^{(i)})=\hat{x}} R(U_{\phi,j}^{(i)}|y^{(i)}).$$

Предложенный алгоритм декодирования требует $O(sL\nu\log\nu)$ операций для расчета логарифмических отношений правдоподобия $R(U_{\phi,j}^{(i)}|y^{(i)})$, и $O(s\nu L\log L)$ операций для расчета глобальных метрик и сортировки путей, т.е. общая сложность составляет $O(Ln\log n)$ операций, как и в случае списочного алгоритма Тала-Варди. Однако основным преимуществом данного алгоритма является возможность выполнения параллельного декодирования с использованием s синхронизируемых ЭСДТВ (синхронизация производится путем обмена комбинированными весами). Каждый ЭСДТВ выполняет $\nu = n/s$ шагов, в то время как списочное декодирование обычного полярного (под)кода требует n шагов. Это обеспечивает значительное снижение общей задержки. Накладные расходы, связанные с синхронизацией ЭСДТВ, могут быть снижены путем использования блочных методов обработки, описанных в [62, 180].

4.5. Декодирование кодов Рида-Соломона

4.5.1. Последовательное декодирование

Расчеты показывают, что в случае кодов Рида-Соломона вероятность ошибки декодирования с помощью метода последовательного исключения оказывается значительно большей по сравнению с другими известными подходами. Для повышения корректирующей способности могут использоваться непосредственные обобщения списочного алгоритма Тала-Варди, описанного в разделе 1.4.4, и последовательного алгоритма декодирования, описанного в разделе 4.1.

Учитывая, что вероятности, задаваемые (2.11), могут быть вычислены со сложностью $O(m2^m)$, получим, что в худшем случае сложность этих алгоритмов равна $O(Ln^2\log^2 n)$, где L — максимальное число проходов списочного или последовательного декодера через каждую фазу. В случае передачи двоичного образа кода вычисление вероятностей $\mathcal{W}_m^{(i)}\left\{u_{0,j}^i|y_0^{n-1}\right\}, 0 \le j < m$, для каждого из путей в кодовом дереве требует не более чем $O(mn\log n)$ операций, а их преобразование согласно (2.12) требует суммарно не более чем $O(n^2)$ операций. Таким образом, сложность этих методов декодирования не превосходит $O(L(n\log^2 n + n^2))$.

Заметим, что предложенный метод может быть использован и для декодирования кодов Рида-Соломона длины $n < 2^m$, в том числе и циклических кодов Рида-Соломона. Можно считать, что такие коды получаются из кодов длины 2^m путем выкалывания $2^m - n$ символов. Соответствующие символы при декодировании могут полагаться стертыми.

4.5.2. Перестановочное декодирование

Из доказательства теоремы 3 видно, что множество замороженных символов, а значит, и вероятность ошибки декодирования с помощью метода последовательного исключения, не зависят от используемого базиса $a_0, \ldots, a_{m-1}$ конечного поля $\mathbb{F}_{2^m}$. Заметим, что различные базисы задают различные перестановки элементов декодируемого вектора. Однако для заданной зашумленной последовательности успешность ее декодирования с использованием метода последовательного исключения и его списочных аналогов может существенно зависеть от используемой перестановки. Это может быть использовано для снижения вероятности ошибки.

Предлагаемый метод перестановочного декодирования состоит в применении вышеописанного алгоритма последовательного декодирования к перестановкам принятой последовательности, задаваемым различными базисами $\mathcal{A}_i = (a_{i,0}, \ldots, a_{i,m-1})$, и выборе наиболее вероятного среди найденных кодовых слов. Более конкретно, сопоставим каждому элементу

принятой последовательности элемент $\beta \in \mathbb{F}_{2^m}$, такой что соответствующий символ кодового слова равен $f(\beta), \deg f(x) < k$. Будем обозначать этот элемент принятой последовательности как y_β. Предлагаемый алгоритм декодирования включает следующие шаги:

1. $i \leftarrow 0, \lambda \leftarrow 0, \widehat{\mathbf{c}} \leftarrow (0, \ldots, 0)$.
2. Построить вектор $z_0^{n-1} : z_{\sum_{s=0}^{m-1} a_{i,s} j_s} = y_j$, где $j = \sum_{s=0}^{m-1} j_s 2^s, j_s \in \{0, 1\}, 0 \le j < n$.
3. Выполнить декодирование вектора z_0^{n-1} с помощью списочного или последовательного алгоритма. Пусть c_0^{n-1} — результат декодирования и $\pi_i = \prod_{j=0}^{n-1} P_{U_0|Y_0}(c_j|z_j)$.
4. Если $\pi_i > \lambda$, положить $\lambda \leftarrow \pi_i, \widehat{\mathbf{c}}_j \leftarrow c_{\sum_{s=0}^{m-1} a_{i,s} j_s}, 0 \le j < n$.
5. $i \leftarrow i + 1$. Если $i < N - 1$, перейти к шагу 2.
6. Возвратить кодовое слово $\widehat{\mathbf{c}}$.

Таким образом, сложность данного метода может быть оценена в худшем случае как $O(LNn^2 \log^2 n)$, или $O(LN(n \log^2 n + n^2))$ в случае передачи двоичного образа кода, где L — максимальный размер списка и N — число используемых базисов.

Средняя сложность может быть существенно снижена за счет остановки декодирования после получения первого кодового слова с вероятностью $\lambda \ge \delta$, где δ — некоторая наперед заданная пороговая величина.

4.5.3. Численные результаты

На рис. 4.11 представлены результаты статистического моделирования для случая передачи двоичного образа кода $(31, 15, 16)$ над $\mathbb{F}_{2^5}$ по аддитивному гауссовскому каналу. Приведены результаты как для метода, предложенного в разделе 4.5.2, с различным размером списка L и числом используемых базисов N, так и для метода Кёттера-Варди (KV) [98] и адаптивного алгоритма распространения доверия в сочетании с мягким алгебраическим декодированием(ABP-KV) [58]. В последнем случае использовались до $N_2 = 155$ перестановок принятой последовательности, для каждой из которых выполнялось до $N_1 = 30$ итераций, состоящих в приведении проверочной матрицы двоичного образа кода к каноническому виду на некоторых $m(n-k)$ позициях, выполнении $It_H = 3$ итераций алгоритма распространения доверия и использовании уточненных оценок надежностей символов в алгоритме Кёттера-Варди, как описано в [58].

Результаты для предложенного алгоритма получены для случая использования базисов $\mathcal{A}_i = A_i(\alpha^0, \alpha^1, \ldots, \alpha^4)^T$, где α — примитивный корень многочлена $x^5 + x^2 + 1$, задаваемых

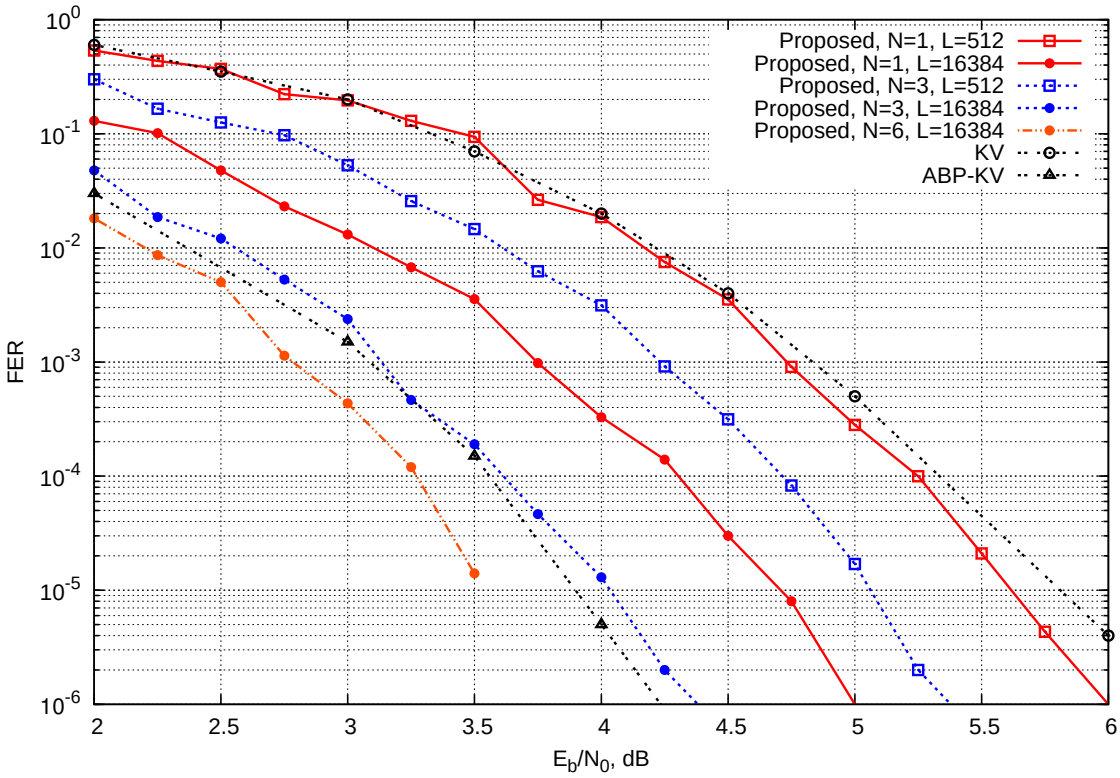


Рис. 4.11. Сравнение различных методов декодирования кода $(31, 15, 16)$

первыми N матрицами A_i из следующего списка[5]:

$$\begin{pmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1 \end{pmatrix}, \begin{pmatrix} 0&0&0&0&1\\ 0&0&0&1&0\\ 0&0&1&0&0\\ 0&1&0&0&0\\ 1&0&0&0&0 \end{pmatrix}, \begin{pmatrix} 0&1&0&1&0\\ 1&0&0&0&0\\ 1&0&0&0&1\\ 1&0&1&0&0\\ 1&1&1&0&1 \end{pmatrix}, \begin{pmatrix} 1&1&1&0&1\\ 1&0&1&0&0\\ 1&0&0&0&1\\ 1&0&0&0&0\\ 0&1&0&1&0 \end{pmatrix}, \begin{pmatrix} 1&1&0&1&1\\ 1&0&0&0&1\\ 0&0&0&0&1\\ 1&0&0&1&0\\ 0&1&1&1&1 \end{pmatrix}, \begin{pmatrix} 0&1&1&1&1\\ 1&0&0&1&0\\ 0&0&0&0&1\\ 1&0&0&0&1\\ 1&1&0&1&1 \end{pmatrix}.$$

Представленные результаты свидетельствуют о том, что предложенный метод при достаточно большом размере списка может обеспечить большую корректирующую способность по сравнению с адаптивным алгоритмом распространения доверия в сочетании с мягким алгебраическим декодированием [58], а также методом Кёттера-Варди [98]. На рис. 4.12 представлена зависимость среднего числа итераций, выполняемых алгоритмом из раздела 4.1, от отношения сигнал/шум для различных N и L. Видно, что с увеличением отношения сигнал/шум средняя сложность предложенного алгоритма быстро уменьшается и стремится к сложности классического алгоритма последовательного исключения.

[5] Данный набор базисов был подобран экспериментально.

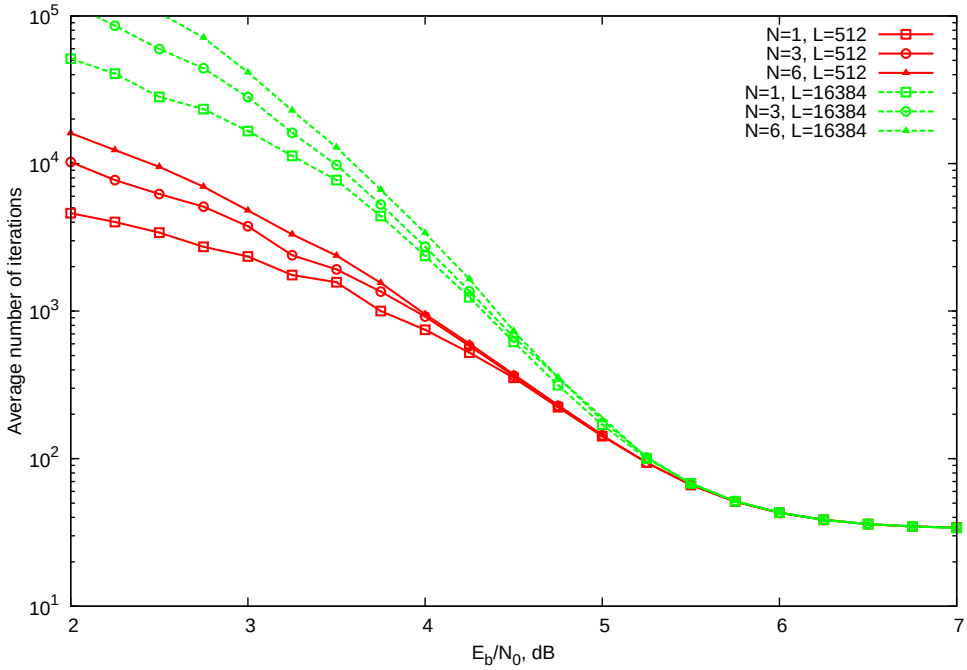


Рис. 4.12. Среднее число итераций алгоритма последовательного декодирования для кода $(31, 15, 16)$

4.6. Выводы

В данной главе был представлен новый последовательный алгоритм декодирования полярных (под)кодов. Было показано, что он имеет существенно меньшую среднюю сложность по сравнению с ранее предложенными стековым и списочным алгоритмами, а также алгоритмами декодирования LDPC и турбо-кодов с сопоставимыми параметрами.

Использование введенного в главе 2 представления линейных блоковых кодов в виде системы ограничений динамического замораживания позволяет использовать предложенный метод для декодирования и других кодов. В частности, как было показано в разделах 2.4.2 и 4.3, в случае коротких кодов РБЧХ предложенный подход позволяет существенно упростить декодирование по сравнению с известным алгоритмом BEAST, а в случае расширенного кода Голея имеет среднюю сложность, сопоставимую со сложностью наилучшего известного алгоритма декодирования этого кода. Кроме того, было представлено обобщение последовательного алгоритма декодирования на случай кодов Рида-Соломона и показано, что для коротких кодов предлагаемый подход обеспечивает лучшую корректирующую способность по сравнению с адаптивным алгоритмом распространения доверия, использующим метод Кётера-Варди. Необходимо отметить, что с увеличением длины кодов в общем случае размер списка, требуемый для обеспечения декодирования почти по максимуму правдоподобия, растет экспоненциально, что связано с ростом числа ненандежных незамороженных символов. В связи с этим, при наличии ограничений на сложность декодирования вероятность ошибки

декодирования с использованием предложенного подхода может превышать таковую для других известных алгоритмов декодирования. Но даже в этом случае данный подход позволяет выполнить декодирование различных кодов с использованием единого инструментария. Это позволяет существенно упростить реализацию систем передачи информации, использующих различные типы корректирующих кодов.

В главе было представлено обобщение списочного алгоритма ПИ на случай цепных кодов и показано, что его корректирующая способность существенно зависит от порядка обхода символов поляризующих преобразований (расписания декодера). Был предложен простой метод построения расписания и показано, что в сочетании с конструкциями кодов, предложенными в разделе 3.2.2, этот метод позволяет получить лучшую корректирующую способность по сравнению с алгоритмом распространения доверия кодов LDPC.

Был представлен параллельный списочный алгоритм декодирования звездных полярных подкодов, который обеспечивает снижение задержки декодирования по сравнению со списочным алгоритмом ПИ для полярных подкодов с одним поляризующим преобразованием.

Глава 5

Списочное декодирование кодов Рида-Соломона

Коды Рида-Соломона являются одним из наиболее широко используемых классов корректирующих кодов. Как правило, для их декодирования используются алгебраические методы, которые способны исправить не более чем $(d-1)/2$ ошибок, где d — минимальное расстояние кода. Предложенный в разделе 4.5 метод имеет приемлемую сложность и корректирующую способность только для коротких кодов Рида-Соломона. Для длинных кодов декодирование с исправлением числа ошибок, большего чем $(d-1)/2$, может быть выполнено с помощью алгоритмов Гурусвами-Судана и Ву. Хотя их сложность является полиномиальной функцией от длины кода, на практике она также оказывается слишком большой для практических целей. Наиболее сложным этапом этих алгоритмов является интерполяционный шаг. В данном разделе представлен быстрый метод, реализующий этот шаг.

Предлагаемый подход описан в разделе 5.1 для случая алгоритма Гурусвами-Судана, и в разделе 5.2 для случая алгоритма Ву.

5.1. Быстрая интерполяция в алгоритме Гурусвами-Судана

5.1.1. Базис модуля интерполяционных многочленов

5.1.1.1. Свойства

Напомним, что интерполяционный полином, требуемый алгоритмом Гурусвами-Судана, может быть найден в базисе Грёбнера модуля $M_{r,\rho} = \left\{Q(x,y) \in I_r \middle| \operatorname{wdeg}_{(0,1)} Q(x,y) < \rho\right\}$ относительно $(1, k-1)$-взвешенного мономиального упорядочения, где I_r — идеал многочленов, имеющих корни кратности r в точках (x_i, y_i).Рассмотрим некоторые свойства базиса Грёбнера $M_{r,\rho}$.

В некоторых случаях будет удобно рассматривать вектор многочленов $A_i(x,y) = \sum_{j=0}^{t} y^j a_{ji}(x), 0 \le i \le s$, как $(1, y, \ldots, y^t) \underbrace{\begin{pmatrix} a_{00}(x) & a_{01}(x) & \ldots & a_{0s}(x) \\ a_{10}(x) & a_{11}(x) & \ldots & a_{1s}(x) \\ \vdots & \vdots & \ddots & \vdots \\ a_{t0}(x) & a_{t1}(x) & \ldots & a_{ts}(x) \end{pmatrix}}_{\mathcal{A}(x)}$, где $\mathcal{A}(x)$ — $(t+1)\times(s+1)$ матричный многочлен.

Лемма 4. *Пусть* $\mathcal{Q} = (Q_0(x,y), \dots, Q_{\rho-1}(x,y))$ *— вектор многочленов, найденных ИИА для исходных данных* $(n, \{(x_i, y_i), 0 \le i < n\}, r, \rho)$*, т.е. базис Грёбнера* $M_{r,\rho}$*. Тогда*

$$\deg\det \mathcal{Q}(x) = \Delta(\mathcal{Q}) = n\frac{r(r+1)}{2},$$

где $\mathcal{Q}(x)$ *— соответствующий матричный многочлен,* $\Delta(\mathcal{Q}) = \sum_{j=0}^{\rho-1} b_j$*, и* $\mathrm{LT}\, Q_j(x,y) = a_j x^{b_j} y^j$*.*

Доказательство. Заметим, что на каждой итерации ИИА происходит увеличение степени относительно x ровно одного многочлена, причем его степень увеличивается ровно на 1. Следовательно, сумма степеней относительно x всех многочленов после завершения работы алгоритма равна числу частных производных Хассе, значения которых были приравнены нулю. С другой стороны, данный алгоритм может быть интерпретирован как построение матричного многочлена

$$\mathcal{Q}(x) = \prod_{i=0}^{n-1} \prod_{\alpha+\beta<r} \delta^{(i,\alpha,\beta)}, \tag{5.1}$$

где

$$\delta^{(i,\alpha,\beta)} = \begin{pmatrix} 1 & 0 & \dots & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ -\frac{\Delta_0}{\Delta_m} & -\frac{\Delta_1}{\Delta_m} & \dots & x - x_i & \dots & -\frac{\Delta_{\rho-1}}{\Delta_m} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & \dots & 1 \end{pmatrix},$$

и $m = m(i,\alpha,\beta)$ — номер наименьшего многочлена, выбираемый на шаге 7 алгоритма. Очевидно, что $\det \delta^{(i,\alpha,\beta)} = \gamma_{i,\alpha,\beta}(x - x_i)$ для некоторого ненулевого $\gamma_{i,\alpha,\beta}$. Число сомножителей в (5.1) также равно числу частных производных Хассе, значения которых были приравнены нулю. □

Лемма 5. *Многочлены* $(S_0(x,y), \dots, S_{s-1}(x,y))$ *образуют базис Грёбнера модуля* $M = \{\sum_{j=0}^{s-1} S_j(x,y)a_j(x) | a_j(x) \in \mathbb{F}[x]\}$*, если все значения* $\mathrm{ydeg}\, S_i(x,y), 0 \le i \le s-1$*, являются различными.*

Доказательство. Лемма вытекает из критерия S-пар Бухбергера [39]. □

Следует отметить, что для любого заданного мономиального упорядочения могут существовать различные базисы Грёбнера рассматриваемого модуля. Они, однако, обладают следующим общим свойством.

Лемма 6. *Пусть* $\mathcal{B} = (B_0(x,y),\dots,B_{\rho-1}(x,y))$ *— некоторый базис Грёбнера* $M_{r,\rho}$. *Тогда* $\Delta(\mathcal{B}) = \frac{nr(r+1)}{2}$.

Доказательство. Пусть $Q_j(x,y), 0 \le j \le \rho-1$ — базис Грёбнера $M_{r,\rho}$, найденный ИИА для того же мономиального упорядочения. Не ограничивая общности, будем считать, что $\mathrm{LT}\, B_j(x,y) = a_j x^{b_j} y^j$. Тогда $\mathrm{LT}\, Q_j(x,y) | \mathrm{LT}\, B_j(x,y)$ и $\mathrm{LT}\, B_j(x,y) | \mathrm{LT}\, Q_j(x,y)$. Это означает, что старшие члены $B_j(x,y)$ и $Q_j(x,y)$ совпадают с точностью до домножения на константы из $\mathbb{F}$. Таким образом, утверждение вытекает из леммы 4. □

5.1.1.2. Преобразование базиса модуля

В [104, 26, 105, 163] было показано, что идеал многочленов I_r порождается

$$\Pi_{r,j}(x,y) = (y - T(x))^j \phi^{r-j}(x), 0 \le j \le r, \tag{5.2}$$

где $T(x_i) = y_i, 0 \le i < n$, и $\phi(x) = \prod_{i=0}^{n-1}(x - x_i)$. Таким образом, базис модуля $M_{r,\rho}$ имеет вид $\mathcal{L} = (\Pi_{r,0}(x,y),\dots,\Pi_{r,r}(x,y),\Pi_{r,r+1}(x,y),\dots,\Pi_{r,\rho-1}(x,y))$, где

$$\Pi_{r,r+j}(x,y) = y^j \Pi_{r,r}(x,y), 0 < j < \rho - r. \tag{5.3}$$

Базис $\mathcal{L}$ необходимо преобразовать в базис Грёбнера относительно $(1, k-1)$-взвешенного лексикографического упорядочения. Это можно сделать с помощью алгоритма, описанного в [26, 105], который может рассматриваться как упрощенная версия алгоритма Бухбергера. Будет удобно описать его здесь в несколько модифицированном виде. Будем считать, что его аргументами являются некоторый многочлен $P(x,y)$, базис Грёбнера $(S_0(x,y),\dots,S_{i-1}(x,y))$ некоторого модуля $M \subset \mathbb{F}[x,y]$, и результатом его работы является базис Грёбнера модуля $M' = \{Q(x,y) + a(x)P(x,y) | Q(x,y) \in M, a(x) \in \mathbb{F}[x]\}$. Алгоритм представлен на рис. 5.1.

Лемма 7. *Пусть* $S_j(x,y), 0 \le j \le i-1$ *— многочлены, удовлетворяющие* $\mathrm{LT}\, S_j(x) = \alpha_j x^{s_j} y^j, \mathrm{wdeg}_{(0,1)} S_j(x,y) < i$. *Тогда алгоритм Reduce строит базис Грёбнера модуля* $M = [S_0(x,y),\dots,S_{i-1}(x,y),P(x,y)]$.

Доказательство. Утверждение вытекает из леммы 5 и обратимости преобразований, выполняемых в этом алгоритме. □

Требуемый базис Грёбнера получается как $\mathcal{S}_{\rho-1}$, где

$$\mathcal{S}_j = Reduce(\mathcal{S}_{j-1}, \Pi_{r,j}(x,y)), \mathcal{S}_0 = (\Pi_{r,0}(x,y)). \tag{5.4}$$

Сложность данного метода равна $O(n^4 k^{-2} r^5)$ [105]. Интересно заметить, что в случае $(1, k-1)$-взвешенного лексикографического упорядочения и $r = 1, \rho = 2$, этот подход сводится

REDUCE$((S_0, \ldots, S_{i-1}), P)$

1 $S_i \leftarrow P$
2 **while** $\exists j : (0 \le j < i) \wedge (\operatorname{ydeg} S_j = \operatorname{ydeg} S_i)$
3 **do if** $\operatorname{LT} S_i | \operatorname{LT} S_j$
4 **then** $W \leftarrow S_j - \frac{\operatorname{LT} S_j}{\operatorname{LT} S_i} S_i$
5 $S_j \leftarrow S_i$
6 $S_i \leftarrow W$
7 **else** $S_i \leftarrow S_i - \frac{\operatorname{LT} S_i}{\operatorname{LT} S_j} S_j$
8 **if** $S_i = 0$
9 **then** $i \leftarrow i - 1$
10 **return** $(S_0, \ldots, S_i)$

Рис. 5.1. Многомерный алгоритм Евклида

к методу декодирования Гао [72, 63], причем функция *Reduce* превращается в расширенный алгоритм Евклида с условием раннего останова. В связи с этим будем называть *Reduce* многомерным алгоритмом Евклида. Этот алгоритм известен также как алгоритм Малдерса-Сторжохана [127].

5.1.2. Двоичный интерполяционный алгоритм

В данном разделе представлен новый интерполяционный алгоритм. Он основан на построении последовательности идеалов и модулей многочленов, имеющих корни (x_i, y_i) возрастающей кратности. Предлагаемый подход может рассматриваться как применение известного двоичного метода возведения в степень к нульмерным идеалам.

5.1.2.1. Интерполяция посредством перемножения идеалов

Основным недостатком метода, задаваемого (5.4), является необходимость работы с многочленами, имеющими общие делители больших степеней. Например, $\mathcal{S}_1 = Reduce((\phi^r(x)), \phi^{r-1}(x)(y - T(x))) = \phi^{r-1}(x) Reduce((\phi(x)), y - T(x))$. Кроме того, в (5.2) требуется возводить многочлены в большие степени. Предлагаемый ниже метод избегает этих двух проблем. Это достигается путем построения базисов Грёбнера модулей многочленов с корнями малой кратности и использования их для построения базисов модулей многочленов с корнями большей кратности.

Лемма 8. *Пусть* $I_r = \{Q(x,y) \in \mathbb{F}[x,y] | Q(x_i,y_i) = 0^r, 0 \le i < n\}$. *Тогда* $I_{r_1+r_2} = I_{r_1} \cdot I_{r_2}$.

Доказательство. $I_{r_1+r_2} = \langle (y - T(x))^j \phi^{r_1+r_2-j}(x), 0 \le j \le r_1 + r_2 \rangle = \langle (y - T(x))^{j_1+j_2} \phi^{r_1-j_1+r_2-j_2}(x), j_1 = 0,\dots,r_1, j_2 = 0,\dots,r_2 \rangle = \langle (y - T(x))^{j_1} \phi^{r_1-j_1}(x), j_1 = 0,\dots,r_1 \rangle \cdot \langle (y - T(x))^{j_2} \phi^{r_2-j_2}(x), j_2 = 0,\dots,r_2 \rangle = I_{r_1} \cdot I_{r_2}$. □

Из этой леммы вытекает, что $I_r = \underbrace{I_1 \cdot I_1 \cdots I_1}_{r\text{раз}} = I_1^r$. Очевидно, что для избежания повторяющихся вычислений можно воспользоваться двоичным методом возведения в степень [96], т.е. вычислить

$$I_r = I_1^r = (\dots((I_1^2 \cdot I_1^{r_{m-1}})^2 \cdot I_1^{r_{m-2}})^2 \cdot I_1^{r_{m-3}} \cdots I_1^{r_1})^2 \cdot I_1^{r_0}$$

где $r = \sum_{j=0}^{m} r_j 2^j, r_m = 1$, $I^2 = I \cdot I$, $I^0 = \mathbb{F}[x,y]$, и $I \cdot \mathbb{F}[x,y] = I$.

Главной задачей, рассматриваемой в данном разделе, является эффективное построение базиса Грёбнера произведения идеалов $I' = \langle P_0(x,y), \dots, P_u(x,y) \rangle$ и $I'' = \langle S_0(x,y), \dots, S_v(x,y) \rangle$. Стандартный подход состоит в вычислении

$$I' \cdot I'' = \langle P_i(x,y) S_j(x,y), 0 \le i \le u, 0 \le j \le v \rangle, \tag{5.5}$$

т.е. попарном перемножении всех элементов базисов рассматриваемых идеалов с последующим нахождением базиса Грёбнера. Это требует $(u+1)(v+1)$ перемножений многочленов от двух переменных, а базис $I' \cdot I''$, получаемый таким образом, обладает крайне высокой избыточностью.

Насколько известно автору, проблема быстрого перемножения идеалов в литературе ранее не рассматривалась, кроме как в [21], где перемножение нульмерных взаимно простых идеалов было сведено к линейной свертке. Однако рассматриваемые идеалы не являются взаимно простыми.

Предлагаемый подход состоит в построении просто вычислимого базиса некоторого подыдеала искомого идеала-произведения с последующей его коррекцией. При этом удобно рассматривать множество элементов искомого идеала с ограниченной степенью относительно переменной y, которое образует модуль.

Лемма 9. *Пусть даны многочлены* $P_j(x,y) : P_j(x_i,y_i) = 0^s, 0 \le i < n, 0 \le j \le m$, *такие что* $\mathrm{LT}\, P_j(x,y) = a_j x^{p_j} y^j$, $\mathrm{wdeg}_{(0,1)} P_j(x,y) \le m$, $p_m = 0$, *и*

$$\Delta((P_0(x,y),\dots,P_m(x,y))) = \frac{ns(s+1)}{2}. \tag{5.6}$$

Тогда $I_s = \langle P_j(x,y), 0 \le j \le m \rangle$, *и многочлены* $P_j(x,y)$ *образуют базис Грёбнера этого идеала.*

Доказательство. В соответствии с леммой 5, многочлены $P_j(x,y)$ образуют базис Грёбнера некоторого модуля. Очевидно, что $\langle P_j(x,y), 0 \le j \le m\rangle \subset I_s$. Предположим, что многочлены $P_j(x,y)$ не образуют базис Грёбнера I_s. Это означает, что существует $S(x,y) \in I_s$: $S(x,y) = \sum_{j=0}^{m} q_j(x,y)P_j(x,y) + R(x,y)$, причем члены $R(x,y)$ не делятся на $\mathrm{LT}\, P_j(x,y)$, т.е. $\mathrm{wdeg}_{(0,1)} R(x,y) < m$ и $\mathrm{LT}\, R(x,y) = \beta x^u y^v, u < p_v$. Заметим, что $R(x,y) \in M_{s,m}$. Это означает, что многочлены $P_j(x,y), 0 \le j \le m$, не образуют базис Грёбнера модуля $M_{s,m}$. Истинный базис Грёбнера этого модуля должен включать меньшие многочлены, т.е. сумма степеней относительно x их старших членов должна быть меньше $\frac{ns(s+1)}{2}$. Но это противоречит лемме 6. Следовательно, $R(x,y) = 0$ и $P_j(x,y)$ образуют базис Грёбнера I_s. □

Вместе с тем, могут существовать базисы Грёбнера I_s, не удовлетворяющие условиям этой леммы.

Пусть $I_{r_1} = \langle P_j(x,y), 0 \le j \le u\rangle, I_{r_2} = \langle S_i(x,y), 0 \le i \le v\rangle$ — идеалы, заданные своими базисами Грёбнера, удовлетворяющими условиям вышеприведенной леммы. Базис Грёбнера их произведения $I_{r_1+r_2}$ может быть найден следующим образом. Пусть $(m'_j, m''_j), 0 \le j \le (u+1)(v+1)-1$ — последовательность различных пар целых чисел, таких что $0 \le m'_j \le u, 0 \le m''_j \le v$, и $\mathrm{LT}\left(P_{m'_j}(x,y)S_{m''_j}(x,y)\right) = \alpha_j x^{q_j} y^j$ при $j \le u+v$. Построим базис

$$\mathcal{Q}_{u+v} = (P_{m'_j}(x,y)S_{m''_j}(x,y), 0 \le j \le u+v) \tag{5.7}$$

некоторого подмодуля $M_{r_1+r_2,u+v+1}$. В соответствии с леммой 5, он является базисом Грёбнера. Можно заметить, что $\Delta(\mathcal{Q}_{u+v}) = \sum_{j=0}^{u+v} q_j \ge \frac{n(r_1+r_2)(r_1+r_2+1)}{2}$.

Рассмотрим последовательность базисов

$$\mathcal{Q}_j = Reduce(\mathcal{Q}_{j-1}, P_{m'_j}S_{m''_j}), j > u+v. \tag{5.8}$$

В ходе работы алгоритма *Reduce* производятся сокращения старших членов ее аргументов, вследствие чего $\Delta(\mathcal{Q}_{j+1}) \le \Delta(\mathcal{Q}_j)$. Как только оказывается, что $\Delta(\mathcal{Q}_j) = \frac{n(r_1+r_2)(r_1+r_2+1)}{2}$, соответствующий базис $\mathcal{Q}_j$ является базисом Грёбнера $I_{r_1+r_2}$.

Лемма 10. *$M_{r_1+r_2,u+v+1}$ порождается $\mathcal{Q}_{(u+1)(v+1)-1}$.*

Доказательство. Рассмотрим $Q(x,y) \in I_{r_1+r_2}$, такой что $\mathrm{wdeg}_{(0,1)} Q(x,y) \le u+v$. Любой такой полином может быть представлен как

$$Q(x,y) = \sum_{j=0}^{u} P_j(x,y) \sum_{i=0}^{v} q_{ji}(x,y)S_i(x,y). \tag{5.9}$$

Внутренняя сумма является элементом I_{r_2}. Т.к. многочлены $S_i(x,y), 0 \le i \le v$, образуют базис Грёбнера модуля $M_{r_2,v+1}$ и идеала I_{r_2}, можно воспользоваться алгоритмом деления полиномов от нескольких переменных и получить $Q(x,y) = \sum_{j=0}^{u} P_j(x,y)\left(\sum_{i=0}^{v-1} w_{ji}(x)S_i(x,y) + S_v(x,y)\sum_{i\ge v} y^{i-v}\tilde{w}_{ji}(x)\right)$. Аналогично, $P(x,y) = \sum_{j=0}^{u} P_j(x,y)\sum_{i\ge v} y^{i-v}\tilde{w}_{ji}(x)$ принадлежит I_{r_1}, и алгоритм деления приводит к $P(x,y) = \sum_{j=0}^{u} w_{jv}(x)P_j(x,y) + P_u(x,y)\sum_{j>u} y^{j-u}w_{jv}(x)$. Следовательно, $Q(x,y) = \sum_{j=0}^{u}\sum_{i=0}^{v} P_j(x,y)S_i(x,y)w_{ji}(x) + P_u(x,y)S_v(x,y)y\overline{w}(x,y)$. Последнее слагаемое в этом выражении равно нулю, т.к. $Q(x,y)$ не содержит членов ax^py^q с $q > u+v$, откуда получаем $M_{r_1+r_2,u+v+1} = [P_j(x,y)S_i(x,y), 0 \le j \le u, 0 \le i \le v]$. □

Из этой леммы вытекает, что для любого подходящего полинома $Q(x,y) \in I_{r_1}I_{r_2}$ можно заменить многочлены $q_{ji}(x,y)$ в (5.9) многочленами от одной переменной $w_{ji}(x)$. Это означает, что последовательность $\mathcal{Q}_j$ действительно сходится к базису искомого модуля. Однако, сходимость оказывается крайне медленной; может потребоваться вычисление большого числа произведений многочленов от двух переменных $P_{m'_j}S_{m''_j}$ с последующим применением алгоритма *Reduce*, прежде чем условие (5.6) будет выполнено. Во многих случаях оказывается даже, что $\mathcal{Q}_{j+1} = \mathcal{Q}_j$, т.е. значительная часть пар (m'_j, m''_j) бесполезна.

В связи с этим предлагается заменить попарные произведения $P_j(x,y)S_i(x,y)$ в (5.5) псевдослучайными линейными комбинациями

$$Q_s(x,y) = \sum_{j=0}^{u}\sum_{i=0}^{v} \chi_{sji}P_j(x,y)S_i(x,y), \tag{5.10}$$

где χ_{sji} являются независимыми случайными величинами, равномерно распределенными в $\mathbb{F}$. Очевидно, что такие многочлены по-прежнему порождают произведение идеалов, если линейное преобразование, задаваемое χ_{sji}, является обратимым, что требует задания по крайней мере $(u+1)(v+1)$ таких многочленов $Q_s(x,y)$. Но оказывается, что в среднем для построения базиса произведения идеалов требуется намного меньшее число таких полиномов. Причиной является то, что $Q_s(x,y)$ зависят от всех попарных произведений $P_j(x,y)S_i(x,y)$, и алгоритм построения базиса Грёбнера (например, *Reduce*) может учесть их одновременно. Этот вопрос будет рассмотрен подробнее в разделе 5.1.2.2.

Вместе с тем, было бы неразумно производить вычисления непосредственно по формуле (5.10), т.к. это требует предварительного вычисления всех попарных произведений $P_j(x,y)S_i(x,y)$. Более эффективный подход состоит в построении последовательности ба-

MERGE$((P_0(x,y),\ldots,P_u(x,y)),(S_0(x,y),\ldots,S_v(x,y)),\Delta_0)$

1 **for** $i \leftarrow 0$ **to** $u+v$
2 **do** $Q_i(x,y) = \min_{0\le j\le v} P_{i-j}(x,y)S_j(x,y)$
3 $\mathcal{Q} = (Q_0(x,y),\ldots,Q_{u+v}(x,y))$
4 **while** $\Delta(\mathcal{Q}) > \Delta_0$
5 **do** $\alpha_i \leftarrow rand(), 0 \le i \le u$
6 $\beta_j \leftarrow rand(), 0 \le j \le v$
7 $Q(x,y) \leftarrow (\sum_{i=0}^{u} \alpha_i P_i(x,y))\,(\sum_{i=0}^{v} \beta_i S_i(x,y))$
8 $\mathcal{Q} \leftarrow$ REDUCE$(\mathcal{Q}, Q(x,y))$
9 **return** $\mathcal{Q}$

Рис. 5.2. Построение базиса Грёбнера идеала $I = JK$ из базисов Грёбнера $J = \langle P_0(x,y),\ldots,P_u(x,y)\rangle$ и $K = \langle S_0(x,y),\ldots,S_v(x,y)\rangle$.

зисов

$$\mathcal{Q}'_{j+1} = Reduce\left(\mathcal{Q}'_j, \left(\sum_{i=0}^{u} \alpha_{ij} P_i(x,y)\right)\left(\sum_{i=0}^{v} \beta_{ij} S_i(x,y)\right)\right),$$

где $j \ge u+v$, и α_{ij}, β_{ij} являются случайными величинами, равномерно распределенными в $\mathbb{F}$. Предлагается также построить начальный базис $\mathcal{Q}'_{u+v} = (Q_0(x,y),\ldots,Q_{u+v}(x,y))$ как $Q_i(x,y) = P_{i-j}(x,y)S_j(x,y)$, причем для каждого i выбирается такое j, что $\mathrm{LT}\, Q_i(x,y) = a_i x^{q_i} y^i$ и величины $q_i, 0 \le i \le u+v$, выбираются минимально возможными. Это позволяет снизить число итераций, выполняемых в многомерном алгоритме Евклида. Рис. 5.2 иллюстрирует предлагаемый подход.

Теорема 5. *Пусть даны базисы Грёбнера* $\mathcal{P} = (P_0(x,y),\ldots,P_u(x,y))$ *и* $\mathcal{S} = (S_0(x,y),\ldots,S_v(x,y))$ *идеалов* I_{r_1} *и* I_{r_2}. *Результатом* $Merge(\mathcal{P},\mathcal{S},n^{\frac{r(r+1)}{2}})$ *является базис Грёбнера* I_r, *где* $r = r_1 + r_2$.

Доказательство. Заметим, что последовательность $\mathcal{Q}'_j$ сходится к базису $M_{r_1+r_2,u+v+1}$, т.к. всегда можно подобрать такие α_{ij} и β_{ij}, что линейное преобразование (5.10), задаваемое $\chi_{sij} = \alpha_{is}\beta_{js}$, окажется обратимым при условии того, что построено достаточно большое число таких полиномов. В соответствии с леммой 7, алгоритм *Reduce* всегда строит базис Грёбнера некоторого модуля. В соответствии с леммой 9, этот базис является базисом Грёбнера I_r. □

Замечание 1. *Merge* не обязательно строит минимальный базис Грёбнера идеала I_r. В частности, несколько построенных им многочленов могут иметь $\mathrm{LT}\, Q_j(x,y) = y^j$. Такие многочлены являются избыточными и должны быть исключены, кроме наименьшего.

Общий интерполяционный алгоритм приведен на рис. 5.3. Для целей списочного декодирования кодов Рида-Соломона в нем должно использоваться $(1, k-1)$-взвешенное лексикографическое упорядочение.

INTERPOLATE$(((x_i, y_i), 1 \le i \le n), r)$

1 $\phi(x) \leftarrow \prod_{i=1}^{n}(x - x_i)$
2 $T(x) \leftarrow \sum_{i=1}^{n} y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$
3 $\mathcal{G} \leftarrow (\phi(x))$
4 $j \leftarrow 0$
5 **repeat**
6 $\quad\quad \mathcal{G} \leftarrow$ REDUCE$(\mathcal{G}, y^j(y - T(x)))$
7 $\quad\quad j \leftarrow j + 1$
8 **until** $\mathrm{LT}\, G_j(x, y) = y^j$
9 $\mathcal{B} \leftarrow \mathcal{G}$
10 Пусть $r = \sum_{j=0}^{m} r_j 2^j, r_j \in \{0, 1\}$
11 $R \leftarrow 1$
12 **for** $j \leftarrow m - 1$ **to** 0
13 **do** $R \leftarrow 2R$
14 $\quad \mathcal{B} \leftarrow$ MERGE$(\mathcal{B}, \mathcal{B}, nR(R+1)/2)$
15 $\quad$ **if** $r_j = 1$
16 $\quad\quad$ **then** $R \leftarrow R + 1$
17 $\quad\quad\quad\quad \mathcal{B} \leftarrow$ MERGE$(\mathcal{B}, \mathcal{G}, nR(R+1)/2)$
18 **return** $\mathcal{B}$

Рис. 5.3. Построение базиса Грёбнера I_r

Заметим, что большинстве практических случаев многочлен $T(x)$ может быть построен с использованием быстрого обратного дискретного преобразования Фурье, которое может также быть использовано для реализации операции умножения многочленов, активно используемой этим алгоритмом.

Теорема 6. *Алгоритм Interpolate строит базис Грёбнера I_r относительно заданного мономиального упорядочения.*

Доказательство. Целью цикла $REPEAT$ является построение базиса Грёбнера модуля $M_{1,j}$, который являлся бы также базисом Грёбнера I_1. Любой базис Грёбнера нульмерного идеала

содержит многочлен $G_j(x,y) : \mathrm{LT}\, G_j(x,y) = ay^j$ для некоторых $a \in \mathbb{F}, j \in \mathbb{N}$ [39, Т. 6.54], поэтому данный цикл завершается за конечное число шагов, и $\mathcal{G}$ действительно является базисом Грёбнера I_1.

Пусть $r' = \sum_{i=j+1}^{m} r_i 2^{i-j-1}$. По индукции, аргументами $Merge$ при вызове из строки 14 являются две копии базиса Грёбнера $I_{r'}$. В соответствии с теоремой 5 результатом его работы является базис Грёбнера $I_{2r'}$. Аналогичные аргументы применимы к строке 17. Следовательно, в конце каждой итерации цикла FOR формируется базис Грёбнера $I_{2r'+r_j}$. Можно заметить, что при этом также выполняется $R = 2r' + r_j$. □

Интерполяционный многочлен, требуемый методом Гурусвами-Судана списочного декодирования, может быть найден как наименьший элемент базиса, формируемого алгоритмом $Interpolate$.

5.1.2.2. Анализ сложности

Оценим скорость сходимости алгоритма $Merge$. Напомним, что он строит базис Грёбнера $M_{r_1+r_2,u+v}$ из базисов Грёбнера $M_{r_1,u}$ и $M_{r_2,v}$ (на самом деле, I_{r_1} и I_{r_2}). Для простоты будем оценивать вероятность того, что $M_{r_1+r_2,u+v}$ порождается $u+v+1+\delta$ многочленами, задаваемыми (5.10), такими что $\mathrm{LT}\, Q_s(x,y) = \gamma_s x^{q_s} y^s$ при $s \le u+v$. Различие в поведении этого идеализированного подхода и вышеописанного алгоритма $Merge$ будут рассмотрены ниже.

Многочлены $Q_0(x,y), \ldots, Q_{u+v+\delta}(x,y)$ могут быть представлены в виде $(u+v+1) \times (u+v+1+\delta)$ многочленной матрицы $\mathcal{Q}(x)$. Если они действительно порождают $M_{r_1+r_2,u+v}$, то $(u+v+1) \times (u+v+1)$ полиномиальная матрица $\mathcal{B}(x)$, соответствующая базису Грёбнера $B_0(x,y), \ldots, B_{u+v}(x,y)$ этого модуля, удовлетворяет

$$\mathcal{Q}(x)\mathcal{A}(x) = \mathcal{B}(x),$$

где $\mathcal{Q}(x)$ — многочленная матрица, соответствующая $Q_i(x,y)$, а $\mathcal{A}(x)$ — матрица, задающая некоторое преобразование. С другой стороны, $Q_j(x,y) \in M_{r_1+r_2,u+v}$, т.е. $\mathcal{Q}(x) = \mathcal{B}(x)\Lambda(x)$, где элементы $\lambda_{ji}(x)$ матрицы $\Lambda(x)$ могут быть получены с помощью алгоритма деления. Следовательно,

$$\mathcal{B}(x)\Lambda(x)\mathcal{A}(x) = \mathcal{B}(x). \tag{5.11}$$

Т.к. $\mathrm{LT}\, B_j(x,y) = x^{b_j} y^j, 0 \le j \le u+v$, многочлены $B_j(x,y)$ линейно независимы над $\mathbb{F}[x]$, и матрица $\mathcal{B}(x)$ обратима над полем рациональных функций, что позволяет сократить ее

в уравнении (5.11). Таким образом, задача сводится к оценке вероятности существования полиномиальной матрицы $\mathcal{A}(x)$, удовлетворяющей

$$\Lambda(x)\mathcal{A}(x) = I. \tag{5.12}$$

Это есть система линейных уравнений относительно $\mathcal{A}(x)$. Заметим, что $\Lambda(x)$ является матрицей полного ранга над $\mathbb{F}[x]$. Многочленное решение этой системы существует тогда и только тогда, когда скалярное матричное уравнение

$$\Lambda(x_w)\mathcal{A}(x_w) = I \tag{5.13}$$

разрешимо для любого $x_w \in \mathbb{F}^*$, т.е. матрицы $\Lambda(x_w)$ имеют ранг $u+v+1$. Достаточно рассмотреть только такие x_w, что некоторая фиксированная $(u+v+1)\times(u+v+1)$-подматрица $\widehat{\Lambda}(x)$ матрицы $\Lambda(x)$ теряет ранг при $x = x_w$, т.е. корни $\det\widehat{\Lambda}(x)$. Такие корни называются собственными числами многочленной матрицы $\widehat{\Lambda}(x)$ [73].

Пусть $\widehat{\Lambda}(x)$ — матрица, состоящая из первых $u+v+1$ столбцов $\Lambda(x)$. Эта матрица удовлетворяет

$$\mathcal{B}(x)\widehat{\Lambda}(x) = \widehat{\mathcal{Q}}(x),$$

где многочленная матрица $\widehat{\mathcal{Q}}(x)$ соответствует $Q_0(x,y),\dots,Q_{u+v}(x,y)$. Для каждого собственного числа $x_w \in \mathbb{F}^*$ многочленной матрицы $\widehat{\Lambda}(x)$ можно найти n_w линейно независимых левых собственных векторов $z^{(w,1)},\dots,z^{(w,n_w)}$, таких что $z^{(w,l)}\widehat{\Lambda}(x_w) = 0, 1 \le l \le n_w$. Геометрическая кратность n_w собственного числа x_w ограничена сверху его алгебраической кратностью r_w, которая определяется как кратность корня x_w определителя $\det\widehat{\Lambda}(x)$. Уравнение (5.13) разрешимо, если для каждого l $\sum_{j=0}^{u+v} z_j^{(w,l)}\lambda_{ji}(x_w) \neq 0$ не менее чем для одного $i : u+v < i \le u+v+\delta$, т.е. если $\Lambda(x_w)$ является матрицей полного ранга. Общее количество таких пар $(x_w, z^{(w,l)})$ ограничено сверху числом $N = \deg\det\widehat{\Lambda}(x) = \deg\det\widehat{\mathcal{Q}}(x) - \deg\det\mathcal{B}(x)$. Многочлены $Q_i(x,y), 0 \le i \le u+v$, представляют собой базис Грёбнера некоторого подмодуля модуля $M_{r_1+r_2,u+v}$ и могут быть получены из многочленов, соответствующих $\mathcal{B}(x)$, путем выполнения действий, определяемых строками 6–10 ИИА для нескольких дополнительных точек (x_i, y_i) и/или пар (α, β). Следовательно, в соответствии с леммой 4, $\Delta_1 = \deg\det\widehat{\mathcal{Q}}(x) = \Delta((Q_0(x,y),\dots,Q_{u+v}(x,y)))$ и $\Delta_0 = \deg\det\mathcal{B}(x) = n\frac{(r_1+r_2)(r_1+r_2+1)}{2}$.

Пусть многочлены $Q_i(x,y), 0 \le i \le w$ образуют базис Грёбнера I_R и $M_{R,w+1}$ относительно $(1, k-1)$-взвешенного лексикографического упорядочения для некоторых R и w. Тогда $\mathrm{LT}\,Q_i(x,y) = x^{q_i}y^i, 0 \le i \le w$, где[1] $q_i \approx l_R - i(k-1), 0 \le i \le w-1$ для некоторого l_R, $q_w = 0$ и

[1] Строгого доказательства этой аппроксимации нет. Однако, можно утверждать, что многочлены, вхо-

$\sum_{i=0}^{w} q_i = n\frac{R(R+1)}{2}$. Тогда $wl_R - (k-1)\frac{w(w-1)}{2} \approx n\frac{R(R+1)}{2}$ и

$$l_R = l_R(w) \approx \frac{(k-1)w(w-1) + nR(R+1)}{2w}.$$

Т.к. многочлены $P_i(x,y)$ и $S_j(x,y)$ образуют базисы Грёбнера модулей $M_{r_1,u}$ и $M_{r_2,v}$, $\mathrm{LT}\, P_i(x,y) = x^{p_i}y_i$, $\mathrm{LT}\, S_j(x,y) = x^{s_j}y^j$, где $p_i \approx l_{r_1}(u) - i(k-1)$ и $s_j \approx l_{r_2}(v) - j(k-1)$. Тогда $q_i \approx l_{r_1}(u) + l_{r_2}(v) - i(k-1), 0 \le i \le u+v$, $q_{u+v} = 0$ и $N = \Delta_1 - \Delta_0 = \sum_{i=0}^{u+v} t_i - n\frac{(r_1+r_2)(r_1+r_2+1)}{2} \approx (l_{r_1}(u) + l_{r_2}(v))(u+v) - (k-1)\frac{(u+v)(u+v-1)}{2} - n\frac{(r_1+r_2)(r_1+r_2+1)}{2}$ Таким образом,

$$N \approx -(k-1)\frac{u+v}{2} + \frac{n}{2uv}\left((vr_1 - ur_2)^2 + v^2 r_1 + u^2 r_2\right). \tag{5.14}$$

Предположим, что элементами $\Lambda(x)$ являются многочлены, коэффициентами которых являются независимые случайные величины, равномерно распределенные на $\mathbb{F}$. Тогда $\overline{\lambda}_{iwl} = \sum_{j=0}^{u+v} z_j^{(w,l)} \lambda_{ji}(x_w)$ — случайные величины, равномерно распределенные на $\mathbb{F}^s$, где $\mathbb{F}^s$ — наименьшее алгебраическое расширение $\mathbb{F}$, такое что $x_w \in \mathbb{F}^s$, и s — степень расширения. В этом случае вероятность того, что $\overline{\lambda}_{iwl}$ отличны от нуля по крайней мере для одного $i \in \{u+v+1, \ldots, u+v+\delta\}$ равна $\theta_{s\delta} = 1 - \frac{1}{|F|^{s\delta}}$.

Рассмотрим факторизацию $\det\widehat{\Lambda}(x) = \alpha\prod_i \phi_i(x)$, где $\alpha \in \mathbb{F}\setminus\{0\}$ и $\phi_i(x) \in \mathbb{F}[x]$ — неприводимые над $\mathbb{F}$ нормированные полиномы. Каждое собственное число x_w является корнем по крайней мере одного $\phi_i(x)$, поэтому $x_w \in \mathbb{F}^{\sigma_i}, \sigma_i = \deg\phi_i(x)$ и $N = \sum_i \sigma_i$. Пусть $\omega_j = |\{i|\sigma_i = j\}|, 1 \le j \le N$. Заметим, что $\phi_i(x)$ имеет σ_i различных корней в $\mathbb{F}^{\sigma_i}$. В наихудшем случае, когда геометрическая и алгебраическая кратности собственных чисел совпадают, получается следующее выражение для вероятности разрешимости уравнения (5.13) для всех x_w:

$$\theta_\delta(\omega) = \prod_{j=1}^{N} \theta_{j\delta}^{j\omega_j} = \prod_{j=1}^{N}\left(1 - \frac{1}{|\mathbb{F}|^{j\delta}}\right)^{j\omega_j}. \tag{5.15}$$

Предполагая, что $\det\widehat{\Lambda}(x)$ является многочленом с независимыми коэффициентами, равномерно распределенными на $\mathbb{F}$, можно оценить вероятность получения разложения $\det\widehat{\Lambda}(x)$ с заданным набором параметров ω_j как $P_\omega = \frac{1}{|\mathbb{F}|^N}\prod_{j=1}^{N}\binom{\mu_j+\omega_j+1}{\omega_j}$ [52], где μ_j — число нормированных неприводимых многочленов степени j. Следовательно, вероятность разрешимости (5.12) равна

$$\Theta(\delta) = \sum_{\omega} \theta_\delta(\omega) P_\omega,$$

дящие в базис Грёбнера $M_{R,w}$, должны иметь *примерно* одинаковую $(1, k-1)$-взвешенную степень, т.к. ИИА, который может быть использован для их построения, всегда увеличивает степень наименьшего полинома. Численные эксперименты подтверждают это утверждение. С другой стороны, если принятая последовательность не является кодовым словом, базис Грёбнера нульмерного идеала I_R должен содержать многочлены с $(1, k-1)$-взвешенной степенью как менее, так и более величины, задаваемой (1.30), а приближенное выражение для l_R, полученное ниже, совпадает с этой границей.

где суммирование ведется по всем разбиениям ω of N.

Точное вычисление значения этого выражения не представляется возможным. Однако, можно заметить, что величина (5.15) определяется в основном первым множителем, в то время как известно, что случайный многочлен с коэффициентами, равномерно распределенными на конечном поле $\mathbb{F}$, имеет в среднем один корень в нем [107]. Следовательно, вероятность неразрешимости (5.12) экспоненциально быстро убывает с ростом δ. Таким образом, для достаточно большого $\mathbb{F}$ можно предположить, что базис Грёбнера $I_{r_1+r_2}$ может быть получен из $u+v+1+\delta, \delta = O(1)$ многочленов, задаваемых (5.10).

Выше был проведен анализ непрактичной версии предложенного рандомизированного алгоритма умножения идеалов. Оказывается, что многочленная матрица, соответствующая многочленам $Q_0(x,y),\ldots,Q_{u+v}(x,y)$, формируемым в строке 2 алгоритма $Merge$, имеет более чем одно собственное число в $\mathbb{F}$ с большой алгебраической кратностью. Вместе с тем, его геометрическая кратность оказывается намного меньше (хотя и больше 1), поэтому алгоритм все же сходится достаточно быстро.

Оценим теперь число итераций алгоритма $Reduce$, вызываемого на строке 8 алгоритма $Merge$. Для этого заметим, что задачей $Reduce$ является уменьшение $(1,k-1)$-взвешенной степени многочленов, построенных на строках 2 и 7 алгоритма $Merge$ с примерно $l_{r_1}+l_{r_2}$ до примерно $l_{r_1+r_2}$, т.е. взаимное сокращение членов со слишком большой $(1,k-1)$-взвешенной степенью. В каждом полиноме примерно $(l_{r_1}+l_{r_2}-l_{r_1+r_2})(u+v)$ членов подлежат сокращению. Общее число сокращаемых одночленов может быть оценено как[2] $\sum_{i=0}^{u+v}(l_{r_1}+l_{r_2}-l_{r_1+r_2})(u+v) = \sum_{i=0}^{u+v}(l_{r_1}+l_{r_2}-i(k-1)-(l_{r_1+r_2}-i(k-1)))(u+v) = N(u+v)$. По крайней мере один член сокращается на каждой итерации $Reduce$. Учитывая (5.14), получим, что число итераций в $Reduce$ равно $O(\tilde{r}(n-\sqrt{nk}))$, где $\tilde{r} = r_1+r_2$. Алгоритм работает с полиномами, имеющими $O(n(2\tilde{r})^2)$ членов, т.е. его сложность равна $O(n(n-\sqrt{nk})\tilde{r}^3)$.

Из (1.29) видно, что число полиномов в базисе $I_{\tilde{r}}, \tilde{r} \le r$ равно $O(\tilde{r}\sqrt{n/k})$. Степени этих полиномов могут быть оценены как $\mathrm{wdeg}_{(0,1)} Q_i(x,y) = O(\tilde{r}\sqrt{n/k})$ и $\mathrm{wdeg}_{(1,0)} Q_i(x,y) = O(n\tilde{r})$. Нахождение произведения таких полиномов требует $O(n\tilde{r}^2\sqrt{n/k}\log(\tilde{r}\sqrt{n/k})\log(n\tilde{r}))$ операций. Вышеприведенный анализ свидетельствует о том, что число итераций, выполняемых $Merge$, равно $O(1)$. Таким образом, умножение многочленов, выполняемое при построении базиса Грёбнера $I_{2\tilde{r}}$ из базиса Грёбнера $I_{\tilde{r}}$, требует $O(\frac{n^2}{k}\tilde{r}^3\log(\tilde{r}\sqrt{n/k})\log(n\tilde{r}))$ операций. Следовательно, общее число операций, выполняемых $Merge$ при вызове со строки 11 интерполяционного алгоритма, составляет $O(n\tilde{r}^3(a\log(\tilde{r}\sqrt{n/k})\log(n\tilde{r})+b(n-\sqrt{nk})))$ для некоторых

[2] Заметим, что целью минимизации на строке 2 алгоритма $Merge$ является уменьшение числа сокращаемых членов, т.к. снижение числа итераций в $Reduce$.

положительных a и b.

Очевидно, что сложность алгоритма $Interpolate$ определяется циклом FOR. Число вызовов $Merge$ в этом цикле равно

$$M = \lfloor \log_2 r \rfloor + \sum_{i=0}^{\lfloor \log_2 r \rfloor} r_i. \tag{5.16}$$

Второй член в этом выражении соответствует строке 17 алгоритма. Сложность всего алгоритма определяется последней его итерацией. Таким образом, общее число операций может быть оценено как $O(nr^3(a\log(r\sqrt{n/k})\log(nr) + b(n-\sqrt{nk})))$. Заметим, что асимптотически это меньше, чем сложность ИИА.

5.1.2.3. Перекодирование

Предложенный алгоритм может быть использован совместно с преобразованием перекодирования [99, 56, 113]. Как было показано в разделе 5.1.1.2, $M_{r,\rho} = [(y-T(x))^j \phi^{r-j}(x), y^s(y-T(x))^r, 0 \le j \le r, 1 \le s \le \rho - r]$. Пусть $\psi(x) = \prod_{i=1}^{k}(x - x_i)$. Деля $T(x)$ на $\psi(x)$ с остатком, получим

$$T(x) = h(x)\psi(x) + g(x),$$

где $g(x_i) = y_i, 1 \le i \le k$ и $h(x_i) = \frac{y_i - g(x_i)}{\psi(x_i)}, i = k+1, \ldots, n$. Выполняя подстановку $y = g(x) + z\psi(x)$ и производя деление[3] всех элементов $M_{r,\rho}$ на $\psi^r(x)$, получим модуль

$$\widehat{M}_{r,\rho} = \left\{ P(x,z) \in \mathbb{F}[x,y] \,\middle|\, P\left(x_i, \frac{y_i - g(x_i)}{\psi(x_i)}\right) = 0^r, \right.$$
$$\left. k+1 \le i \le n, \operatorname{wdeg}_{(0,1)} P(x,z) < \rho \right\},$$

который порождается $\widehat{\Pi}_{r,j}(x,z) = (z-h(x))^j \theta^{r-j}(x), 0 \le j \le r$ и $\widehat{\Pi}_{r,r+j} = z^j \psi^j(x)(z-h(x))^r, 1 \le j \le \rho - r$, где $\theta(x) = \frac{\phi(x)}{\psi(x)}$. Имеет место взаимно однозначное соответствие между многочленами в $M_{r,\rho}$ и $\widehat{M}_{r,\rho}$, причем наименьший многочлен относительно $(1, k-1)$-взвешенного лексикографического упорядочения в $M_{r,\rho}$ соответствует наименьшему многочлену относительно $(1,-1)$-взвешенного лексикографического упорядочения[4] в $\widehat{M}_{r,\rho}$. Если старший член некоторого полинома в $M_{r,\rho}$ имеет вид ax^iy^j, то старший член соответствующего полинома в $\widehat{M}_{r,\rho}$ равен $ax^{i+jk-rk}z^j, a \in \mathbb{F}$.

Базис Грёбнера $\widehat{M}_{r,\rho}$ может быть получен с помощью модифицированной версии алгоритма $Interpolate$, приведенной на рис. 5.4; при этом должно использоваться $(1,-1)$-взвешенное лексикографическое упорядочение. Вначале данный алгоритм строит ба-

[3] Данная операция не позволяет использовать здесь понятие идеала.

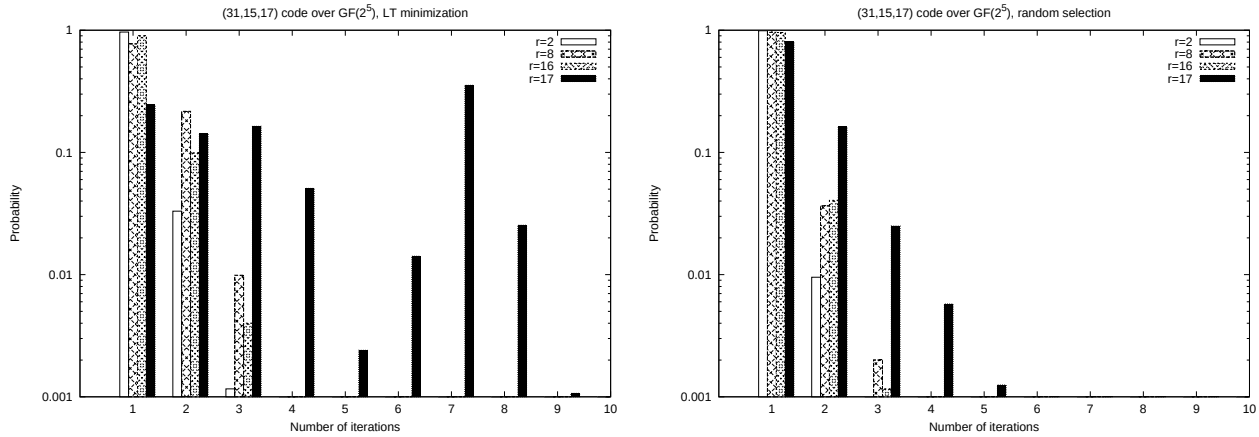
[4] Заметим, что последнее не является в строгом смысле мономиальным упорядочением, т.к. относительно него невозможно определить наименьший одночлен.

REENCODEINTERPOLATE$(((x_i, y_i), 1 \le i \le n), r, k)$

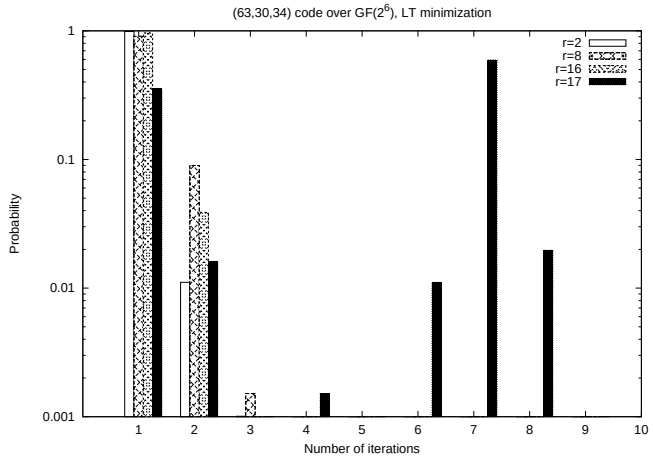
1 $\psi(x) \leftarrow \prod_{i=1}^{k}(x - x_i); \theta(x) = \prod_{i=k+1}^{n}(x - x_i)$
2 $T(x) \leftarrow \sum_{i=1}^{n} y_i \frac{\prod_{j\ne i}(x-x_j)}{\prod_{j\ne i}(x_i-x_j)}$
3 Compute $h(x) : T(x) = h(x)\psi(x) + g(x), \deg g(x) < k$
4 $\mathcal{G} \leftarrow (\theta(x)); j \leftarrow 0$
5 **repeat**
6 $\mathcal{G} \leftarrow$ REDUCE$(\mathcal{G}, (\psi(x)z)^j(z - h(x)))$
7 $j \leftarrow j + 1$
8 **until** $\mathrm{LT}\, G_j = x^{(j-1)k} z^j$
9 $\mathcal{B} \leftarrow \mathcal{G}; R \leftarrow 1$
10 Пусть $r = \sum_{j=0}^{m} r_j 2^j, r_j \in \{0, 1\}$
11 **for** $j \leftarrow m - 1$ **to** 0
12 **do** $R \leftarrow 2R$
13 $\mathcal{B} \leftarrow$ MERGE$(\mathcal{B}, \mathcal{B}, \delta(|\mathcal{B}|, |\mathcal{B}|, R))$
14 **if** $r_j = 1$
15 **then** $R \leftarrow R + 1$
16 $\mathcal{B} \leftarrow$ MERGE$(\mathcal{B}, \mathcal{G}, \delta(|\mathcal{B}|, |\mathcal{G}|, R))$
17 **return** $\mathcal{B}$

Рис. 5.4. Построение базиса Грёбнера $\widehat{M}_{r,\rho}$

зис Грёбнера $\widehat{M}_{1,j+1}$. Соответствующий цикл исходной версии алгоритма останавливается сразу же после получения многочлена со старшим членом y^j. После замены переменных и упорядочения условие останова преобразуется в $\mathrm{LT}\, G_j = x^{(j-1)k} z^j$. Далее производится наращивание кратности корней тем же способом, что и в первоначальной версии алгоритма. При этом, однако, порог останова Δ_0 алгоритма $Merge$ должен быть изменен. Для кратности корней R, после замены переменных базисные многочлены имеют старшие члены $x^{\widehat{t_i}} z^i, 0 \le i \le u-1$, такие что $\widehat{t_i} = t_i + (i-R)k$, где $x^{t_i} y^i$ — старшие члены многочленов, которые были бы получены без перекодирования. Таким образом, порог останова в модифицированном алгоритме должен быть установлен равным $\widehat{\Delta}_0 = \sum_{i=0}^{u-1} (t_i + (i - R)k) = \Delta_0 - Rku + k\frac{(u-1)u}{2} = \Delta_0 - k\frac{R(R+1)}{2} + k\frac{(u-1-R)(u-R)}{2}$, где Δ_0 — порог, полученный из условия (5.6). Если размеры базисов, переданных в качестве аргументов $Merge$, равны u_1 и u_2, то $u = u_1 + u_2 - 1$. Таким образом, порог останова $Merge$ должен быть положен равным $\delta(u_1, u_2, R) = (n - k)\frac{R(R+1)}{2} + k\frac{(u_1+u_2-2-R)(u_1+u_2-1-R)}{2}$.



(*а*) (*б*)



(*в*)

Рис. 5.5. Распределение числа итераций в алгоритме $Merge$

5.1.3. Численные результаты

В данном разделе представлены результаты статистического моделирования, иллюстрирующие эффективность предложенного подхода. Все предложенные алгоритмы были реализованы на языке C++, при этом использовался алгоритм Карацубы быстрого умножения многочленов [96] для реализации шагов 2 и 7 алгоритма $Merge$.

Рис. 5.5 иллюстрирует распределение числа итераций, выполняемых алгоритмом $Merge$ при построении базиса Грёбнера идеала I_r для различных значений r (декодирование по Гурусвами-Судану). Рассматривались коды $(31, 15, 17)$ над $GF(2^5)$ и $(63, 30, 34)$ над $GF(2^6)$. Первая и последняя гистограммы были получены с использованием алгоритма, представленного на рис. 5.2. Вторая гистограмма соответствует алгоритму, полученному путем замены операции минимизации старших членов на строке 2 алгоритма случайным выбором $Q_i(x, y)$ из множества $\{P_i(x, y)S_0(x, y), \ldots, P_{i-v}(x, y)S_v(x, y)\}$. Можно заметить, что в этом случае алгоритм действительно демонстрирует экспоненциальную сходимость. В первом и третьем

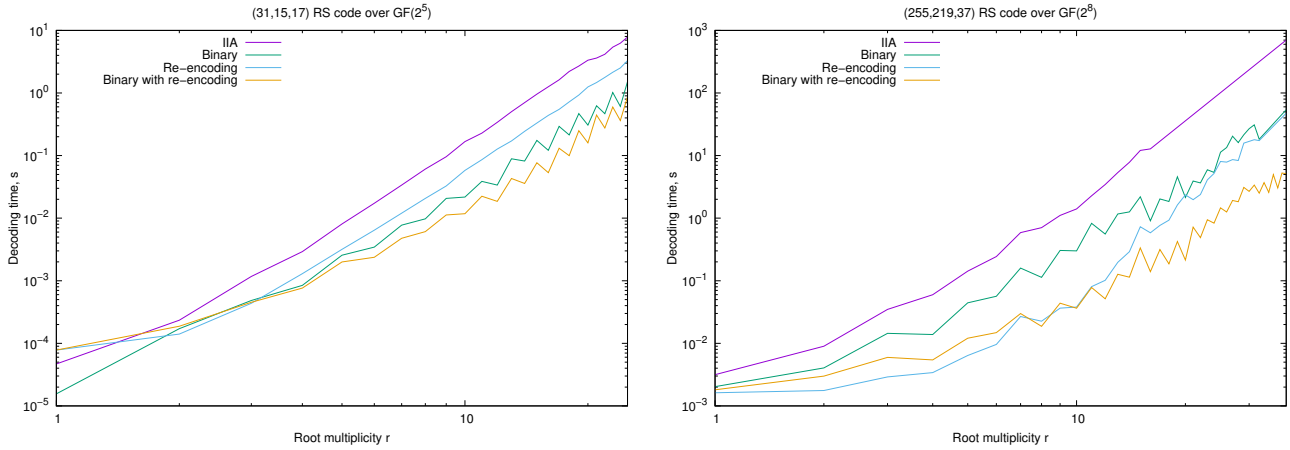


Рис. 5.6. Время декодирования с помощью различных алгоритмов

случаях сходимость остается экспоненциальной при $r = 2^m$, хотя и оказывается несколько более медленной по сравнению с методом случайного выбора начального базиса. Но при $r = 17$ с большой вероятностью требуется до 8 итераций. Это объясняется тем, что построение базиса Грёбнера идеала $I_{17} = I_{16} \cdot I_1$ требует обработки двух различных наборов многочленов $P_i(x, y), 0 \le i \le u$ и $S_j(x, y), 0 \le j \le v$, являющихся базисами I_{16} и I_1 соответственно, причем $v \ll u$. С большой вероятностью наименьший из полиномов $S_{j_0}(x, y)$ используется почти для всех i на строке 2 алгоритма $Merge$. В этом случае коэффициенты $\det \widehat{\Lambda}(x)$ не могут рассматриваться как статистически независимые. В частности, наиболее вероятными разбиениями ω в (5.15) оказываются те из них, у которых ω_1 достаточно велика. Это приводит к появлению у $\widehat{\Lambda}(x)$ собственных чисел в $\mathbb{F}$ с большой алгебраической и геометрической кратностью. В результате требуется большее количество многочленов коррекции (т.е. итераций цикла $WHILE$) для их исключения. Этот эффект частично компенсируется снижением общего числа итераций, выполняемых алгоритмом $Reduce$. Заметим, что при $r = 2^m$ в случае $|\mathbb{F}| = 64$ алгоритм сходится быстрее, чем при $|\mathbb{F}| = 32$, как и вытекает из (5.15).

На рис. 5.6 представлены кривые, показывающие время декодирования с использованием ИИА, алгоритма Ли-О'Салливана, предложенного двоичного интерполяционного алгоритма, ИИА с преобразованием перекодирования, а также двоичного интерполяционного алгоритма с перекодированием. Рассматривались коды Рида-Соломона $(31, 15, 17)$ и $(255, 219, 37)$. Видно, что предлагаемый подход обеспечивает снижение сложности до 12 раз по сравнению с ИИА в случае кода $(31, 15, 17)$, и до 15 раз в случае кода $(255, 219, 37)$. Заметим, что время декодирования с использованием предложенного метода возрастает медленнее, чем в случае ИИА, подтверждая таким образом вывод, сделанный в разделе 5.1.2.2. Заметим также, что в некоторых случаях увеличение кратности корней приводит к снижению сложности предло-

женного интерполяционного метода, что объясняется влиянием второго слагаемого в (5.16), т.е. строки 17 предложенного алгоритма.

Заметим также, что предложенный алгоритм демонстрирует меньшую сложность по сравнению с методом перекодирования и ИИА для низкоскоростных кодов. Для высокоскоростных кодов ИИА, использующий преобразование перекодирования, оказывается проще. Однако применяя преобразование перекодирования совместно с предложенным алгоритмом удается получить дополнительное снижение сложности. Общий выигрыш по сравнению с ИИА достигает 22 раз в случае кода $(31, 15, 17)$ и 157 раз в случае кода $(255, 219, 37)$.

5.2. Быстрая интерполяция в алгоритме By

5.2.1. Описание алгоритма By в частотной области

Проблема списочного декодирования кодов Рида-Соломона может быть сформулирована как поиск всех пар $(f(x), \sigma(x))$, где $\deg f(x) < k$ и $\sigma(x_i) = 0$ для не более чем t различных $x_i : f(x_i) \neq y_i$, so that $y_i\sigma(x_i) = f(x_i)\sigma(x_i), i = 1, \ldots, n$. Здесь t — радиус декодирования, $f(x)$ — многочлен, соответствующий кодовому слову, и $\sigma(x)$ — многочлен ошибок. Заметим, что $\sigma(x)$ может быть найден из $f(x)$ и y_1^n и наоборот.

Любая такая пара полиномов $f(x), \sigma(x)$ может быть представлена в виде многочлена от двух переменных

$$Q(x, y) = y\sigma(x) - f(x)\sigma(x). \tag{5.17}$$

Этот многочлен имеет n корней (x_i, y_i). Следовательно, он принадлежит модулю $\mathcal{M} = [\phi(x), y - T(x)]$, где $\phi(x) = \prod_{i=1}^{n}(x - x_i)$, и $T(x)$ — такой многочлен[5], что $T(x_i) = y_i$. Пусть многочлены $Q'(x, y) = q_{00}(x) + yq_{10}(x)$ и $Q''(x, y) = q_{01}(x) + yq_{11}(x)$ также образуют базис этого модуля. Тогда любой $Q(x, y) \in \mathcal{M}$ может быть представлен как

$$Q(x, y) = a(x)Q'(x, y) + b(x)Q''(x, y). \tag{5.18}$$

Пусть $Q'(x, y)$ и $Q''(x, y)$ образуют базис Грёбнера $\mathcal{M}$ относительно $(1, k-1)$-взвешенного лексикографического упорядочения $y \prec x$, причем $\operatorname{ydeg} Q'(x, y) = 0$ и $\operatorname{ydeg} Q''(x, y) = 1$. В силу свойств базисов Грёбнера, каждый многочлен $Q(x, y)$, имеющий корни (x_i, y_i), может быть представлен в виде (5.18), где $\operatorname{LT} Q(x, y) = \operatorname{LT} a(x) \operatorname{LT} Q'(x, y)$ или $\operatorname{LT} Q(x, y) = \operatorname{LT} b(x) \operatorname{LT} Q''(x, y)$, и многочлены $a(x)$, $b(x)$ могут быть найдены с помощью алгоритма

[5] Заметим, что для произвольного $T(x)$, удовлетворяющего этим условиям, можно найти интерполяционный многочлен наименьшей степени как $\tilde{T}(x) \equiv T(x) \bmod \phi(x)$. Так как $\phi(x) \in \mathcal{M}$, получим $\mathcal{M} = [\phi(x), y - T(x)] = [\phi(x), y - \tilde{T}(x)]$

деления многочленов от нескольких переменных [39]. Это означает, что $\deg b(x) \le w_2 = t - \deg q_{11}(x)$, и $\deg a(x) \le w_1 = t + k - 1 - \deg q_{00}(x)$. Таким образом, списочное декодирование в коде Рида-Соломона может быть выполнено путем перебора всех таких многочленов $a(x), b(x)$, построения соответствующих многочленов $Q(x, y)$ и проверки возможности их разложения в соответствии (5.17), с учетом того, что:

1. $\sigma(x)$ должен быть допустимым многочленом ошибок, т.е. он должен иметь ровно $\deg \sigma(x)$ корней среди $\{x_1, \ldots, x_n\}$.

2. $f(x)$ должен совпадать с $(y_1, \ldots, y_n)$ в достаточно большом числе позиция.

Таким образом, алгоритм включает следующие шаги:

1. Построить $T(x) : T(x_i) = y_i, i = 1, \ldots, n$.

2. Найти многочлены $Q'(x, y) = q_{00}(x) + yq_{10}(x)$ и $Q''(x, y) = q_{01}(x) + yq_{11}(x)$, образующие базис Грёбнера $\mathcal{M} = [\phi(x), y - T(x)]$ относительно $(1, k-1)$-взвешенного лексикографического упорядочения. Данный шаг аналогичен применению расширенного алгоритма Евклида с условием раннего останова, лежащего в основе метода Гао [72].

3. [Рациональная интерполяция] Найти все пары взаимно простых многочленов $a(x), b(x) : \deg a(x) \le w_1 = t + k - 1 - \deg q_{00}(x), \deg b(x) \le w_2 = t - \deg q_{11}(x)$, причем

$$\sigma(x) = a(x)q_{10}(x) + b(x)q_{11}(x) \tag{5.19}$$

   должен иметь не более t корней.

4. Для каждого найденного $\sigma(x)$ восстановить кодовое слово по таким символам y_i, что x_i не являются корнями $\sigma(x)$.

Описанный алгоритм может рассматриваться как интерпретация метода Ву в частотной области [193]. Он был независимо получен в [28].

Напомним, что в изначально метод Ву был описан на основе аналитического продолжения алгоритма Берлекэмпа-Месси и предполагал поиск всех таких пар многочленов $(a(x), b(x))$, что

$$\Lambda^*(x) = a(x)\Lambda(x) + xB(x)b(x) \tag{5.20}$$

имеет не более t различных корней α^{-j_i}, где $j_i, 1 \le i \le t$ — позиция i-ой ошибки, а $\Lambda(x)$ и $B(x)$ — многочлены, построенные алгоритмом Берлекэмпа-Месси из вектора синдрома. Применение языка базисов Грёбнера делает описание метода Ву намного проще.

5.2.2. Интерполяция частично однородными многочленами

Идея метода By состоит в замене исчерпывающего перебора многочленов $a(x), b(x)$ на алгебраическую процедуру, аналогичную методу Гурусвами-Судана. Более конкретно, для заданных многочленов $\Lambda(x)$ и $B(x)$ [см. (5.20)], соответствующие пары многочленов $a(x), b(x)$ могут быть найдены путем построения многочлена $S(x, y)$, имеющего корни $(x_i, -\frac{\Lambda(x_i^{-1})}{x_i^{-1}B(x_i^{-1})})$ кратности r для некоторого $r \geq 1$, и поиска решения уравнения $a^\rho(x)S(x, b(x)/a(x)) = 0$, где ρ — степень $S(x, y)$ относительно переменной y. Может оказаться так, что $B(x_i^{-1}) = 0$ для некоторых i, т.е. некоторые интерполяционные точки могут иметь вторую компоненту, равную бесконечности. По определению, $S(x, y)$ имеет корень (x_i, ∞) тогда и только тогда, когда $y^{\deg_y S(x,y)}S(x, y^{-1})$ имеет корень $(x_i, 0)$. Однако многие существующие интерполяционные алгоритмы [129], [105] не могут быть непосредственно использованы для нахождения $S(x, y)$, т.к. они ориентированы на построение базиса идеала многочленов с заданными корнями, в то время как искомые многочлены не образуют идеал.

Действительно, многочлены $1 - xy$ и $1 - x^2y$ имеют корень $(0, \infty)$. Но многочлен $x(1 - xy) - (1 - x^2y) = x - 1$ не имеет такого корня.

Такая же проблема возникает и в переформулированном алгоритме, основанном на (5.19).

Описанное выше техническое затруднение может быть разрешено путем введения частично однородных многочленов $S(x, y, z) = \sum_{j=0}^{\rho} s_j(x)z^{\rho-j}y^j$. Следующая лемма устанавливает связь между предлагаемым подходом и исходным методом By.

Лемма 11. *Пусть $S(x, y, z) = \sum_{j=0}^{\rho}\sum_i s_{ji}x^iy^jz^{\rho-j}$ — многочлен, однородный относительно переменных y и z. Он имеет корни кратности r в точках $(x_0, \alpha y_0, -\alpha z_0)$ для всех $\alpha \in \mathbb{F}$, где y_0 и z_0 одновременно не равны нулю, тогда и только тогда, когда*

1. *$\hat{S}(x, \theta) = \sum_{j=0}^{\rho}\sum_i s_{ji}x^i\theta^j$ имеет корень $(x_0, -y_0/z_0)$ кратности r (для $z_0 \neq 0$);*
2. *$\tilde{S}(x, \theta) = \sum_{j=0}^{\rho}\sum_i s_{\rho-j,i}x^i\theta^j$ имеет корень $(x_0, -z_0/y_0)$ кратности r (для $y_0 \neq 0$).*

Доказательство. Предположим, не ограничивая общности, что $z_0 \neq 0$. $S(x, y, z) = \sum_{j=0}^{\rho}\sum_i s_{ji}x^iy^jz^{\rho-j}$ имеет корни кратности r в точках $(x_0, \alpha y_0, -\alpha z_0)$ тогда и только тогда, когда его производные Хассе в этих точках общего порядка менее r равны нулю, т.е.,

$$\sum_{i'\geq u}\sum_{j'=v}^{\rho-w}\binom{i'}{u}\binom{j'}{v}\binom{\rho-j'}{w}s_{j'i'}x_0^{i'-u}\frac{(\alpha y_0)^{j'-v}}{(-\alpha z_0)^{j'+w-\rho}} = 0$$

для всех таких $u, v, w \geq 0$, что. $u + v + w < r$. Тогда при $w = 0$ получим

$$(-z_0)^{\rho-v}\sum_{i'\geq u}\sum_{j'=v}^{\rho}\binom{i'}{u}\binom{j'}{v}s_{j'i'}x_0^{i'-u}\left(-\frac{y_0}{z_0}\right)^{j'-v} = 0$$

для всех $u, v : u + v < r$, т.е. $(x_0, -y_0/z_0)$ является корнем кратности r многочлена $\hat{S}(x, \theta) = \sum_{j=0}^{\rho} \sum_i s_{ji} x^i \theta^j$.

Если $(x_0, -y_0/z_0)$ является корнем $\hat{S}(x, \theta)$ кратности r, то

$$\hat{S}(x, \theta) = \sum_{\substack{u+v \geq r \\ v \leq \rho}} s^{[u,v]} (x - x_0)^u (\theta + y_0/z_0)^v.$$

Следовательно,

$$S(x, y, z) = z^\rho \hat{S}(x, y/z) = \sum_{\substack{u+v \geq r \\ v \leq \rho}} \frac{s^{[u,v]}}{(z_0)^v} (x - x_0)^u (yz_0 + zy_0)^v z^{\rho - v} =$$

$$= \sum_{\substack{u+v \geq r \\ v \leq \rho}} \frac{s^{[u,v]}}{(z_0)^v} (x - x_0)^u ((y - \alpha y_0) z_0 + (z + \alpha z_0) y_0)^v z^{\rho - v}.$$

Можно заметить, что $S(x + x_0, y + \alpha y_0, z - \alpha z_0)$ не содержит членов общей степенью меньше r для любых α, т.е. точки $(x_0, \alpha y_0, -\alpha z_0)$ являются его корнями кратности r. □

Далее представлены переформулированные леммы 5 и 6 из [193].

Лемма 12. *Пусть $S(x, y, z) = \sum_{j=0}^{\rho} s_j(x) y^j z^{\rho - j}$ — многочлен, имеющий корень кратности r в точках $(x_0, \alpha y_0, -\alpha z_0)$ для любых α, причем y_0 и z_0 одновременно не обращаются в ноль. Если $a(x), b(x)$ — многочлены, удовлетворяющие $z_0 a(x_0) + y_0 b(x_0) = 0$, то $(x - x_0)^r | S(x, a(x), b(x))$.*

Доказательство. Предположим, не ограничивая общности, что $z_0 \neq 0$. Тогда в силу леммы 11 получим, что $\widehat{S}(x, \theta) = \sum_{j=0}^{\rho} s_j(x) \theta^j$ имеет корень $(x_0, -y_0/z_0)$ кратности r, т.е.,

$$\widehat{S}(x, \theta) = \sum_{\substack{u+v \geq r \\ v \leq \rho}} \widehat{s}^{[u,v]} (x - x_0)^u (\theta + y_0/z_0)^v.$$

Следовательно,

$$S(x, y, z) = \sum_{\substack{u+v \geq r \\ v \leq \rho}} \frac{\widehat{s}^{[u,v]}}{z_0^v} (x - x_0)^u (yz_0 + zy_0)^v z^{\rho - v}.$$

Утверждение леммы вытекает из того, что x_0 является корнем $z_0 a(x) + y_0 b(x)$. □

Лемма 13. *Пусть $S(x, y, z) = \sum_{j=0}^{\rho} s_j(x) y^j z^{\rho - j}$ — такой многочлен, что $\operatorname{wdeg}_{(1, w_1, w_2)} S(x, y, z) < rt$ и точки $(x_i, \alpha y_i, -\alpha z_i), i = 1, \ldots, n$, где y_i и z_i одновременно не обращаются в ноль, являются его корнями кратности r для всех α. Еслм $a(x)$ и $b(x)$ — многочлены такие, что $\deg a(x) \leq w_1, \deg b(x) \leq w_2$ и $z_i a(x_i) + y_i b(x_i) = 0$ для не менее чем t точек (x_i, y_i, z_i), то $S(x, a(x), b(x)) = 0$.*

Доказательство. В силу леммы 12, для любой такой точки $(x - x_i)^r | S(x, a(x), b(x))$. Степень $g(x) = S(x, a(x), b(x))$ не превосходит $\operatorname{wdeg}_{(1, w1, w2)} S(x, y, z) < rt$. Таким образом, единственной возможностью для многочлена $\prod_i (x - x_i)^r$ степени rt быть делителем $g(x)$ является $S(x, a(x), b(x)) = 0$. □

Ограничения на кратность корней задают $nr(r+1)/2$ линейных уравнений. Эту систему уравнений можно решить и получить искомый многочлен, если число неизвестных в ней превосходит число уравнений, т.е., $\sum_{j=0}^{\rho}(rt - jw_1 - (\rho - j)w_2) = rt(\rho+1) - w\frac{\rho(\rho+1)}{2} > n\frac{r(r+1)}{2}$, где $w = w_1 + w_2$. При $w = 0$ это означает $r = 1$ и

$$\rho > n\frac{r+1}{2t} - 1 = \frac{n}{t} - 1. \tag{5.21}$$

При $w > 0$ получим

$$\rho_l = \frac{2rt - w - \sqrt{D}}{2w} < \rho < \frac{2rt - w + \sqrt{D}}{2w} = \rho_h \tag{5.22}$$

где $D = (w + 2rt)^2 - 4wnr(r+1) > 0$. Из последнего неравенства вытекает

$$r > \frac{\left(n - t + \sqrt{n^2 - 2tn + wn}\right)w}{2(t^2 - wn)}. \tag{5.23}$$

Это неравенство может быть разрешено при $t^2 - wn \geq 0$. Т.к. базис Грёбнера $\mathcal{M}$ удолвлетворяет $\deg q_{11}(x) + \deg q_{00}(x) = n$, получим, что $w = w_1 + w_2 = 2t + (k-1) - n$. Таким образом, декодирование возможно при $t < n - \sqrt{n(k-1)}$.

Как правило, r должно выбираться наименьшим возможным с целью минимизации сложности. Но для малых r может оказаться, что интервал, задаваемый (5.22), не содержит ни одного целого. В этом случае необходимо увеличить r до тех пор, пока не будет найдено подходящее значение ρ. Гарантировать его существование можно при $\rho_h - \rho_l = \sqrt{D}/w > 1$. Простые вычисления приводят к

$$r > \frac{w(n-t)}{t^2 - wn}.$$

Теорема 7. *Пусть $\overline{y} = (y_1, \ldots, y_n)$ — некоторый вектор из $\mathbb{F}^n$. Рассмотрим многочлены $Q'(x,y) = q_{00}(x) + yq_{10}(x), Q''(x,y) = q_{01}(x) + yq_{11}(x)$, образующие базис Грёбнера модуля $\mathcal{M} = [\phi(x), y - T(x)]$ относительно $(1, k-1)$-взвешенного лексикографического упорядочения, где $T(x) : T(x_i) = y_i, 1 \leq i \leq n, \phi(x) = \prod_{i=1}^{n}(x - x_i)$. Если $t < n - \sqrt{n(k-1)}$ и параметры r, ρ удовлетворяют (5.21)–(5.23), то все кодовые слова $\overline{c} = (c_1, \ldots, c_n)$ $(n, k, n-k+1)$ кода РС над $\mathbb{F}$, удовлетворяющие $d_H(\overline{c}, \overline{y}) = t' \leq t$, могут быть найдены из многочленов $\sigma(x) = a(x)q_{10}(x) + b(x)q_{11}(x)$, причем $c_i \neq y_i \Leftrightarrow \sigma(x_i) = 0$, $S(x, a(x), b(x)) = 0$. Здесь $S(x,y,z) = \sum_{i=0}^{\rho} s_i(x)y^i z^{\rho-i}$ является таким многочленом, что для всех $\alpha \in \mathbb{F}$ точки $(x_i, \alpha q_{11}(x_i), -\alpha q_{10}(x_i)), 1 \leq i \leq n$, являются его корнями кратности r, и $\operatorname{wdeg}_{(1,w_1,w_2)} S(x,y,z) < rt$, где $w_1 = t + k - 1 - \deg q_{00}(x), w_2 = t - \deg q_{11}(x)$.*

Доказательство. Условия (5.21)-(5.23) гарантируют возможность построения ненулевого многочлена $S(x,y,z)$. Для каждого кодового слова $\overline{c}$ существуют многочлены $f(x), \sigma(x)$, удовлетворяющие $\deg f(x) < k, f(x_i) = \overline{c}_i, 1 \leq i \leq n$, $\sigma(x) = \prod_{i:c_i \neq y_i}(x - x_i)$. Если $d_H(\overline{c}, \overline{y}) = t' \leq t$, то

$\sigma(x) = a(x)q_{10}(x)+b(x)q_{11}(x)$, где $\deg a(x) \le t'+k-1-\deg q_{00}(x)$ и $\deg b(x) \le t'-\deg q_{11}(x)$. Пусть $\omega(x)$ — многочлен наименьшей степени, имеющий $t-t'$ корней на множестве локаторов неошибочных символов $\{x \in \{x_1,\dots,x_n\} | \overline{c}_i = y_i\}$. Тогда $\sigma'(x) = \sigma(x)\omega(x) = a'(x)q_{10}(x) + b'(x)q_{11}(x)$ имеет t различных корней и $\deg a'(x) \le t+k-1-\deg q_{00}(x), \deg b'(x) \le t-\deg q_{11}(x)$. Из леммы 13 вытекает, что $0 = S(x,a'(x),b'(x)) = S(x,\omega(x)a(x),\omega(x)b(x)) = \omega^\rho(x)\sum_{j=0}^{\rho} s_j(x)a^j(x)b^{\rho-j}(x) = \omega^\rho(x)S(x,a(x),b(x))$, т.е., $S(x,a(x),b(x)) = 0$. □

Для заданного многочлена $S(x,y,z)$, удовлетворяющего вышеприведенным условиям, многочлены $a(x), b(x)$ могут быть восстановлены модифицированным алгоритмом Рота-Рукенштейн, описанным в [193]. При этом оказывается, что ρ задает верхнюю границу на размер списка, т.е. число различных многочленов ошибок $\sigma(x)$, которые могут быть найдены описанным способом.

Для того, чтобы изучить поведение полученных границ, введем нормированную корректирующую способность $\tau = t/n$ и скорость[6] кода $R = (k-1)/n$. Тогда (5.23) может быть переписана как

$$r > \left\lceil \frac{(1-\tau+\sqrt{R})(2\tau+R-1)}{2(\tau^2-2\tau-R+1)} \right\rceil.$$

Аналогично,

$$\rho_l = \frac{r\tau - \sqrt{(\tau(r+1)+\frac{R-1}{2})^2-(2\tau+R-1)r(r+1)}}{2\tau+R-1} - \frac{1}{2}.$$

Эти оценки применимы и к исходному алгоритму Ву.

На рис. 5.7 представлено сравнение этой границы и границы, полученной в [193] ($\rho_{Wu} = \left\lfloor \frac{rt}{w} \right\rfloor = \left\lfloor \frac{r\tau}{2\tau+R-1} \right\rfloor$). Можно заметить, что новая граница существенно лучше, в особенности для малых τ. Кроме того, ее поведение намного более естественно, т.е. размер списка увеличивается с увеличением радиуса декодирования.

5.2.3. Эффективная интерполяция

5.2.3.1. Частично однородные многочлены

Как было показано выше, все пары $(a(x),b(x))$, для которых многочлен $\sigma(x) = a(x)q_{10}(x) + b(x)q_{11}(x)$ имеет t различных корней, задаются уравнением $S(x,a(x),b(x)) = 0$, где $S(x,y,z)$ — многочлен, имеющий корни $(x_i, \alpha q_{11}(x_i), -\alpha q_{10}(x_i))$ кратности r и $(1,w_1,w_2)$-взвешенную степень менее rt. Этот многочлен может быть найден в базисе Грёбнера идеала I_r многочленов, имеющих такие корни [151]. Однако полный базис Грёбнера этого идеала содержит множество многочленов, не удовлетворяющих (5.22), которые

[6] Здесь используются обозначения, введенные в [80].

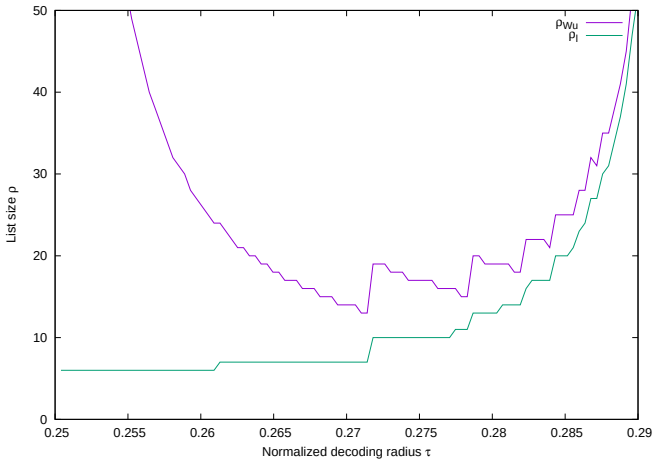


Рис. 5.7. Сравнение границ на размер списка

являются бесполезными для целей декодирования. Достаточно лишь рассмотреть подмодуль $M_{\rho,r} = \left\{S(x,y,z) \in I_r | S(x,y,z) = \sum_{j=0}^{\rho} s_j(x) z^{\rho-j} y^j\right\}$ этого идеала и его базис Грёбнера $Q_0(x,y,z), \ldots, Q_\rho(x,y,z)$. Любой $Q(x,y,z) \in M_{\rho,r}$ может быть представлен как $S(x,y,z) = \sum_{j=0}^{\rho} Q_j(x,y,z) p_j(x)$. Один из многочленов $Q_j(x,y,z)$ гарантированно удовлетворяет ограничению на взвешенную степень.

Искомый базис Грёбнера может быть найден с помощью итеративного интерполяционного алгоритма (ИИА) [129], если его этап инициализации заменить на $Q_j(x,y,z) := z^{\rho-j} y^j$. Этот подход требует $O(n^2 r^4 \rho)$ операций. Т.к. (5.23) позволяет использовать намного меньшие r по сравнению с алгоритмом Гурусвами-Судана, описанный подход обеспечивает значительное снижение сложности. Вместе с тем, она остается слишком высокой для практических целей.

Дальнейшее снижение сложности может быть достигнуто путем обобщения вышеописанного двоичного интерполяционного алгоритма на случай частично однородных многочленов. Основная идея предлагаемого подхода состоит в том, чтобы вначале построить базис модуля многочленов малых степеней с небольшой кратностью корней, а затем нарастить кратность корней и степень многочленов относительно y, z. Следующая лемма задает начальную точку для этой последовательности модулей.

Лемма 14. *Пусть $q_{11}(x)$ и $q_{10}(x)$ — взаимно простые многочлены. Тогда $M_{1,1} = \widehat{M}$, где $\widehat{M} = [\phi(x)z, \phi(x)y, q_{11}(x)z + q_{10}(x)y]$.*

Доказательство. Построим вначале линейно независимые над $\mathbb{F}[x]$ многочлены, порождающие модуль $\widehat{M}$, и затем покажем, что они действительно порождают $M_{1,1}$.

С помощью расширенного алгоритма Евклида можно найти такие многочлены

$u_{00}(x), u_{10}(x), u_{01}(x), u_{11}(x)$, что

$$g_{11}(x) = \gcd(\phi(x), q_{10}(x)) = u_{10}(x)\phi(x) + u_{11}(x)q_{10}(x) \tag{5.24}$$

и

$$0 = u_{00}(x)\phi(x) + u_{01}(x)q_{10}(x). \tag{5.25}$$

Пусть

$$\begin{aligned}\tilde{G}_0(x,y,z) &= u_{00}(x)\phi(x)y + u_{01}(x)(q_{11}(x)z + q_{10}(x)y)\\ &= u_{01}(x)q_{11}(x)z\\ G_1(x,y,z) &= u_{10}(x)\phi(x)y + u_{11}(x)(q_{11}(x)z + q_{10}(x)y)\\ &= u_{11}(x)q_{11}(x)z + g_{11}(x)y.\end{aligned}$$

Эти многочлены совместно с $\phi(x)z$ образуют базис $\widehat{M}$. Введем также многочлен $G_0(x,y,z) = \gcd(\phi(x), u_{01}(x)q_{11}(x))z$. Видно, что $u_{01}(x)q_{11}(x) = -\frac{\phi(x)u_{00}(x)q_{11}(x)}{q_{10}(x)} = -\frac{\phi(x)}{g_{11}(x)}\frac{q_{11}(x)u_{00}(x)}{q'_{10}(x)}$, где $q_{10}(x) = g_{11}(x)q'_{10}(x)$. Из (5.25) вытекает, что $q'_{10}(x)|u_{00}(x)$, т.е., $u_{00}(x) = q'_{10}(x)u'_{00}(x)$. Тогда из (5.25) вытекает, что $u'_{00}(x)\phi(x) = -u_{01}(x)g_{11}(x)$. Из скойств расширенного алгоритма Евклида вытекает, что $u_{00}(x)u_{11}(x) - u_{10}(x)u_{01}(x) = (-1)^i$ для некоторого i. Таким образом,

$$u'_{00}(x)\left(q'_{10}(x)u_{11}(x) + u_{10}(x)\frac{\phi(x)}{g_{11}(x)}\right) = (-1)^i. \tag{5.26}$$

Следовательно, $u'_{00}(x) = \pm 1$ и $u_{01}(x)q_{11}(x) = \pm\frac{\phi(x)}{g_{11}(x)}q_{11}(x)$. Тогда

$$G_0(x,y,z) = \frac{\phi(x)}{g_{11}(x)}\gcd(g_{11}(x), q_{11}(x))z = \frac{\phi(x)}{g_{11}(x)}z. \tag{5.27}$$

Последнее равенство следует из взаимной простоты $q_{11}(x)$ и $q_{10}(x)$. Т.к. преобразования, использованные для получения $G_0(x,y,z)$ и $G_1(x,y,z)$ из $\phi(x)z, \phi(x)y$ и $q_{11}(x)z - q_{10}(x)y$, обратимы, эти многочлены порождают тот же модуль $\widehat{M}$.

Пусть $A(x,y,z) = u(x)z + v(x)y$ — многочлен из $M_{1,1}$, т.е. $u(x)q_{10}(x) - v(x)q_{11}(x) = a(x)\phi(x)$ для некоторого $a(x)$. Т.к. $g_{11}(x)|q_{10}(x), g_{11}(x)|\phi(x)$ и $\gcd(q_{11}(x), q_{10}(x)) = 1$, $v(x)$ делится на $g_{11}(x)$. Рассмотрим многочлен

$$\begin{aligned}R(x,y,z) &= A(x,y,z) - \frac{v(x)}{g_{11}(x)}G_1(x,y,z)\\ &= z\left(u(x) - \frac{v(x)u_{11}(x)q_{11}(x)}{g_{11}(x)}\right)\\ &= z\left(u(x) - \frac{u_{11}(x)}{g_{11}(x)}(u(x)q_{10}(x) - a(x)\phi(x))\right)\\ &= z\left(u(x)\left(1 + \frac{u_{10}(x)\phi(x) - g_{11}(x)}{g_{11}(x)}\right) + \frac{a(x)u_{11}(x)\phi(x)}{g_{11}(x)}\right)\\ &= z\frac{\phi(x)}{g_{11}(x)}\left(u_{10}(x)u(x) + a(x)u_{11}(x)\right).\end{aligned}$$

Этот многочлен делится на $G_0(x,y,z)$. Таким образом, произвольный многочлен $A(x,y,z) \in M_{1,1}$ может быть выражен через $G_0(x,y,z), G_1(x,y,z)$, базисные многочлены $\widehat{M}$. Следовательно $M_{1,1} \subset \widehat{M}$. Включение $\widehat{M} \subset M_{1,1}$ следует из определения $M_{1,1} \subset I_1$. □

Искомый интерполяционный многочлен может быть найден в $M_{\rho,r}$, где параметры ρ и r должны удовлетворять ограничениям, полученным в разделе 5.2.2. Предлагаемый подход состоит в построении последовательности модулей $M_{u,v}$ (и их базисов Грёбнера), получаемых с помощью следующих операций:

1. расширение модуля, которая строит $M_{u+1,v}$ из $M_{u,v}$;
2. увеличение кратности корней, которая строит $M_{u'+u'',v'+v''}$ из $M_{u',v'}$ и $M_{u'',v''}$.

Эти операции рассматриваются ниже.

5.2.3.2. Расширение модуля

Следующая лемма указывает полезное свойство базисов Грёбнера $M_{\rho,r}$ относительно лексикографического ($y \prec z \prec x$) мономиального упорядочения.

Лемма 15. *Пусть $Q_0(x,y,z),\dots,Q_\rho(x,y,z)$ — базис Грёбнера $M_{\rho,r}$ относительно лексикографического упорядочения, где $\rho \geq r$. Тогда $Q_\rho(x,y,z) = g_{11}^r(x)y^\rho + Q'(x,y,z)$, где $g_{11}(x)$ задается (5.24) и $Q'(x,y,z)$ не делится на y^ρ.*

Доказательство. Многочлены в рассматриваемом базисе имеют вид $Q_i(x,y,z) = \sum_{j=0}^{i} q_{ji}(x)y^j z^{\rho-j}$. Единственным многочленом среди них, делящимся на y^ρ, является $Q_\rho(x,y,z) = \sum_{j=0}^{\rho} q_{j,\rho}(x)y^j z^{\rho-j}$. Он имеет корни $(x_i, \alpha q_{11}(x_i), -\alpha q_{10}(x_i))$ кратности r. Из этого следует, что для всех таких i, что $q_{10}(x_i) = g_{11}(x_i) = 0$, многочлен $\tilde{Q}_\rho(x,\theta) = \sum_{j=0}^{\rho} q_{\rho-j,\rho}(x)\theta^j$ имеет корни $(x_i, 0)$ кратности r. Следовательно, $(x-x_i)^r | q_{\rho,\rho}(x)$. Т.к. $g_{11} = \prod_{i:q_{1,1}(x_i)=0}(x-x_i)$, получим, что $g_{11}^r | q_{\rho,\rho}(x)$. С другой стороны, $z^{\rho-r}G_1^r(x,y,z) \in M_{\rho,r}$, где $G_1(x,y,z) = u_{11}(x)q_{11}(x)z + g_{11}(x)y$. Т.к. $Q_0(x,y,z),\dots,Q_\rho(x,y,z)$ является базисом Грёбнера относительно лексикографического упорядочения, $q_{\rho,\rho}(x) | g_{11}^r(x)$. Следовательно, $q_{\rho,\rho}(x) = g_{11}^r(x)$. □

Этот результат аналогичен известному свойству нульмерных идеалов кольца $\mathbb{F}[x,y]$, любой базис Грёбнера которых должен содержать многочлены $Q'(x,y), Q''(x,y)$, удовлетворяющие $\mathrm{LT}\, Q'(x,y) = y^a, \mathrm{LT}\, Q''(x,y) = x^b$.

Следующая лемма описывает простое свойство, которое позволяет выявить факт получения базиса Грёбнера рассматриваемого модуля.

EXPAND$((S_0(x,y,z),\dots,S_\rho(x,y,z)),n,r)$

1 $\mathcal{G}_0 \leftarrow (zS_0(x,y,z),\dots,zS_\rho(x,y,z),yS_\rho(x,y,z))$
2 $j \leftarrow 1$
3 **while** $\Delta(\mathcal{G}_{j-1}) > n\frac{r(r+1)}{2}$
4 **do** $\beta_{ij} \leftarrow \mathrm{RAND}(), 0 \le i \le \rho$
5 $P_j(x,y,z) \leftarrow y\sum_{i=0}^{\rho}\beta_{ij}S_i(x,y,z)$
6 $\mathcal{G}_j \leftarrow \mathrm{REDUCE}(\mathcal{G}_{j-1}, P_j(x,y,z))$
7 $j \leftarrow j+1$
8 **return** $\mathcal{G}_{j-1}$

Рис. 5.8. Построение базиса Грёбнера $M_{\rho+1,r}$ на основе базиса Грёбнера $M_{\rho,r}$

Лемма 16. *Пусть* $Q_j(x,y,z), j = 0,\dots,\rho$ — *такие многочлены, что* $Q_j(x_i,\alpha q_{11}(x_i),-\alpha q_{01}(x_i)) = 0^r$, *и* $\mathrm{ydeg}\,Q_j(x,y,z) = j, j = 0,\dots,\rho$. *Если* $\Delta((Q_0(x,y,z),\dots,Q_\rho(x,y,z))) = n\frac{r(r+1)}{2}$, *то эти многочлены образуют базис Грёбнера* $M_{\rho,r}$.

Доказательство. Доказательство аналогично лемме 9. □

Основным отличием метода By от метода Гурусвами-Судана является необходимость контроля не только кратности корней r, но и степени многочленов ρ. Приведенный ниже алгоритм включает две операции: увеличение как r, так и ρ, и увеличение только ρ. Реализация последней операции основана на следующей лемме.

Лемма 17. *Рассмотрим модуль* $M_{\rho,r} = [Q_0(x,y,z),\dots,Q_\rho(x,y,z)]$. *Тогда* $M_{\rho+1,r} = [zQ_0(x,y,z),\dots,zQ_\rho(x,y,z),yQ_0(x,y,z),\dots,yQ_\rho(x,y,z)]$.

Доказательство. Не ограничивая общности, будем считать, что $Q_0(x,y,z),\dots,Q_\rho(x,y,z)$ образуют базис Грёбнера $M_{\rho,r}$ относительно лексикографического упорядочения. Многочлены $yQ_0(x,y,z),\dots,yQ_\rho(x,y,z)$ порождают некоторый подмодуль $M_{\rho+1,r}$. Всякий многочлен $A(x,y,z) \in M_{\rho+1,r}$ может быть представлен как $A(x,y,z) = a_{\rho+1}(x)y^{\rho+1} + zA'(x,y,z)$, где $A'(x,y,z)$ не содержит членов, делящихся на $y^{\rho+1}$. В силу леммы 15, $g_{11}^r(x)|a_{\rho+1}(x)$, и $yQ_\rho(x,y) = g_{11}^r(x)y^{\rho+1} + \dots$. Таким образом, деля $A(x,y,z)$ на $yQ_0(x,y,z),\dots,yQ_\rho(x,y,z)$, получим остаток $zR(x,y,z)$, где $R(x,y,z) \in M_{\rho,r}$. Следовательно, существуют $q_0(x),\dots,q_\rho(x)$: $zR(x,y,z) = \sum_{j=0}^{\rho} zQ_j(x,y,z)q_j(x)$. □

Следует отметить, что базис, описанный в вышеприведенной лемме, крайне избыточен, т.к. не более $\rho+2$ элементов из $M_{\rho+1,r}$ могут быть линейно независимы над $\mathbb{F}[x]$. Эту проблему можно преодолеть, используя рандомизированный подход, предложенный в разделе 5.1.2.1 для целей умножения идеалов. Более конкретно, предлагается построить последовательность модулей $M^{(j)}_{\rho+1,r} = \left\{P(x,y,z) + a(x)P_j(x,y,z) | a(x) \in \mathbb{F}[x], P(x,y,z) \in M^{(j-1)}_{\rho+1,r}\right\}$, где $M^{(0)}_{\rho+1,r} = [zQ_0(x,y,z),\dots,zQ_\rho(x,y,z), yQ_\rho(x,y,z)]$, многочлены $P_j(x,y,z)$ вычисляются как $P_j(x,y,z) = y\sum_{i=0}^{\rho}\beta_{ij}Q_j(x,y,z)$, β_{ij} — независимые случайные величины, равномерно распределенные на $\mathbb{F}$, и $Q_j(x,y,z)$ — элементы базиса $M_{\rho,r}$. Этот процесс может быть остановлен сразу при выполнеии условий леммы 16. На рис. 5.8 представлен предлагаемый алгоритм. Описанный подход требует вычисления на каждом шаге базиса Грёбнера $(S_0(x,y,z),\dots,S_{\rho+1}(x,y,z))$ модуля $M^{(j)}_{\rho+1,r}$. Это может быть выполнено с помощью многомерного алгоритма Евклида, представленного на рис. 5.1, которые может быть очевидным образом обобщен на случай частично однородных многочленов. Этот алгоритм предполагает, что $\operatorname{ydeg} S_j(x,y,z) = j, 0 \le j < i$, и строит базис Грёбнера с таким же свойством.

Теорема 8. *Пусть дан базис Грёбнера $M_{\rho,r}$, удовлетворяющий $\operatorname{LT} S_i(x,y,z) = z^{\rho-i}y^i x^{u_i}$. На основе него алгоритм Expand строит базис Грёбнера $M_{\rho+1,r}$ со средней сложностью $O(n^2r^4/\rho)$.*

Доказательство. Будем следовать подходу, использованному в разделе 5.1.2.2.

Исходный базис $\mathcal{G}_0$, формируемый на строке 1 алгоритма, удовлетворяет требованиям к исходным данным многомерного алгоритма Евклида, поэтому на каждой итерации он действительно порождает базис Грёбнера $M^{(j)}_{\rho+1,r}$. После выполнения достаточно большого числа итераций, линейное преобразование, задаваемое случайными величинами β_{ij}, становится обратимым, что дает возможность восстановить все многочлены $yS_i(x,y,z)$ из $P_j(x,y,z)$. Таким образом, алгоритм действительно сходится к базису Грёбнера $M_{\rho+1,r}$.

Для оценки скорости сходимости вычислим вероятность того, что $M_{\rho+1,r}$ порождается $zS_i(x,y,z), yS_\rho(x,y,z), P_{i''}(x,y,z), 0 \le i' \le \rho, 1 \le i'' \le \delta$. Эти многочлены могут быть представлены как $(z^{\rho+1}, z^\rho y,\dots,y^{\rho+1})\mathbf{G}(x)$, где $\mathbf{G}(x)$ — $(\rho+1)\times(\rho+1+\delta)$ многочленная матрица. Базис Грёбнера может быть представлен аналогичным образом в виде $(\rho+1)\times(\rho+1)$ матрицы $\mathbf{B}(x)$, удовлетворяющей $\mathbf{G}(x)\mathbf{U}(x) = \mathbf{B}(x)$ для некоторой многочленной матрицы $\mathbf{U}(x)$. С другой стороны, вышеуказанные многочлены принадлежат $M_{\rho+1,r}$, поэтому $\mathbf{G}(x) = \mathbf{B}(x)\mathbf{W}(x)$. Это означает, что $\mathbf{W}(x)\mathbf{U}(x) = I$, т.е. для любого x_0 из $\tilde{\mathbb{F}}$, алгебраического замыкания $\mathbb{F}$, $\mathbf{W}(x_0)$ должна быть матрицей полного ранга. Пусть $\mathbf{W}(x) = (\tilde{\mathbf{W}}(x)|\widehat{\mathbf{W}}(x))$, где $\tilde{\mathbf{W}}(x)$ — $(\rho+2)\times(\rho+2)$ подматрица, состоящая из первых $\rho+2$ столбцов $\mathbf{W}(x)$. Можно найти не

более $N = \deg\det\tilde{\mathbf{W}}(x)$ пар собственных чисел $x_i \in \mathbb{F}^{\sigma_i}$, где $\mathbb{F}^{\sigma_i}$ — некоторое алгебраическое расширение $\mathbb{F}$, и соответствующих линейно независимых собственных векторов $v_i \in (\mathbb{F}^{\sigma_i})^{\rho+1}$, для которых $v_i\mathbf{W}(x_i) = 0$. Полагая элементы $\mathbf{W}(x_i)$ независимыми случайными величинами, равномерно распределенными в $\mathbb{F}^{\sigma_i}$, получим выражение для вероятности того, что $\mathbf{W}(x)$ является матрицей полного ранга для всех x

$$\Theta(\delta) = \sum_{\omega} P_\omega \prod_{j=1}^{N}\left(1 - \frac{1}{|\mathbb{F}|^{j\delta}}\right)^{j\omega_j}$$

где суммирование ведется по всем таким разбиениям ω, что $\sum_j j\omega_j = N$, и P_ω — вероятность существования разложения $\det\tilde{\mathbf{W}}(x)$ на неприводимые над $\mathbb{F}$ многочлены $\phi_s(x)$, из которых ровно ω_j имеют степень j, т.е. имеющие корни x_i в $\mathbb{F}^j$. Значение этого выражения определяется в основном сомножителями, соответствующими $j = 1$, поэтому вероятность того, что $M_{\rho+1,r}$ не порождается $\rho+1+\delta$ многочленам убывает экспоненциально быстро с ростом δ, т.е. *Expand* сходится за $O(1)$ итерацию.

Заметим, что $\det\tilde{\mathbf{W}}(x) = \frac{\det\tilde{\mathbf{G}}(x)}{\det\mathbf{B}(x)}$, т.е., $N = \delta(\mathcal{G}_0)) - n\frac{r(r+1)}{2}$, где $\tilde{\mathbf{G}}(x)$ соответствует начальной аппроксимации $\mathcal{G}_0$, построенной на строке 1. Предполагая, что все многочлены $S_i(x,y,z)$ в первоначальном базисе Грёбнера $M_{\rho,r}$ имеют одинаковую $(1,w_1,w_2)$-взвешенную степень C, получим, что их старшие члены удовлетворяют $\mathrm{LT}\,S_i(x,y,z) = z^{\rho-i}y^i x^{u_i}$, т..е. $u_i + iw_1 + (\rho-i)w_2 = C$ и $\sum_{i=0}^{\rho} u_i = n\frac{r(r+1)}{2}$. Отсюда следует, что $C = (w_1+w_2)\frac{\rho}{2} + n\frac{r(r+1)}{2(\rho+1)}$ и $u_i = (w_1-w_2)\frac{\rho}{2} + n\frac{r(r+1)}{2(\rho+1)} + i(w_2-w_1)$. Таким образом, $\Delta(\mathcal{G}_0) = \sum_{i=0}^{\rho} u_i + u_\rho = n\frac{r(r+1)}{2} + n\frac{r(r+1)}{2(\rho+1)} + \frac{\rho}{2}(w_2-w_1)$, т.е. $N = O(nr^2/\rho)$. В ходе работы цикла $WHILE$, взвешенная степень многочленов в $\mathcal{G}_j$ уменьшается с C до $C' = (w_1+w_2)\frac{\rho+1}{2} + n\frac{r(r+1)}{2(\rho+2)}$. Каждый многочлен содержит не более $\rho+2$ одночленов любой фиксированной $(1,w_1,w_2)$-взвешенной степени между C и C'. Таким образом, общее число членов, которые должны быть сокращены, равно $n\frac{r(r+1)}{2(\rho+1)} - \frac{w_1+w_2}{2}(\rho+2)$. Полагая, что на каждой итерации *Reduce* сокращается ровно один такой член путем сложения соответствующих полиномов, содержащих $O(nr^2)$ членов, общая сложность *Expand* может быть оценена как $O(n^2r^4/\rho)$. □

5.2.3.3. Наращивание кратности корней

Предлагаемый подход повторяет метод, описанный в разделе 5.1.2.1 для случая алгоритма Гурусвами-Судана.

Лемма 18. *Пусть* $M_{\rho_1,r_1} = [S_0(x,y,z),\ldots,S_{\rho_1}(x,y,z)]$ *и* $M_{\rho_2,r_2} = [P_0(x,y,z),\ldots,P_{\rho_2}(x,y,z)]$ *— модули, задаваемые своими базисами Грёбнера, удовлетворяющими условиям леммы 16. Тогда*

$$M_{\rho_1+\rho_2,r_1+r_2} = [S_i(x,y,z)P_j(x,y,z), i = 0..\rho_1, j = 0..\rho_2]. \tag{5.28}$$

Доказательство. См. доказательство леммы 10. □

Эта лемма позволяет обобщить двоичный метод перемножения идеалов на случай задачи рациональной интерполяции. Более конкретно, вместо попарных произведений в (5.28) можно использовать достаточно большое число многочленов

$$Q_j(x,y,z) = \left(\sum_{i=0}^{\rho_1} \alpha_{ij} S_i(x,y,z)\right)\left(\sum_{i=0}^{\rho_2} \beta_{ij} P_i(x,y,z)\right),$$

где α_{ij}, β_{ij} — случайные величины, равномерно распределенные на $\mathbb{F}$. Тогда последовательность $\mathcal{M}^{(j+1)} = \{Q(x,y,z) + a(x)Q_j(x,y,z) | Q(x,y,z) \in \mathcal{M}^{(j)}\}$, где $\mathcal{M}^{(0)} \subset M_{\rho_1+\rho_2, r_1+r_2}$, сходится к $M_{\rho_1+\rho_2, r_1+r_2}$. Целесообразно построить начальный подмодуль $\mathcal{M}^{(0)}$ каким-то простым способом. Например, он может быть определен как модуль, порождаемый многочленами $S_{i-j_i}(x,y,z)P_{ji}(x,y,z), i = 0,\ldots,\rho_1+\rho_2$, где j_i выбираются так, чтобы старший член в получаемом произведении был минимальным.

На рис. 5.9 представлен алгоритм, реализующий этот подход. Необходимо установить $\Delta_0 = n^{\frac{r(r+1)}{2}}, r = r_1 + r_2$, чтобы цикл $WHILE$ завершился при достижении $\Delta(B) = \Delta_0$. Это условие означает, что модуль $\mathcal{M}^{(j)}$, задаваемый построенным базисом Грёбнера $\mathcal{B}$, равен $M_{\rho_1+\rho_2, r_1+r_2}$. Методом, аналогичным использованному в разделе 5.1.2.2, можно показать, что при $r_1 = r_2 = r, \rho_1 = \rho_2 = \rho$ этот алгоритм сходится за $O(1)$ итерацию. Число операций, требуемых для перемножения двух многочленов, равно $O(\rho n r \log\rho \log(nr))$, а среднее число итераций, выполняемых *Reduce* равно $nr\frac{\rho-r}{\rho+1}$, поэтому средняя сложность равна $O(n^2 r^3 \frac{\rho-r}{\rho+1})$.

MERGE$((S_i(x,y,z), i = 0..\rho_1), (P_i(x,y,z), i = 0..\rho_2), \Delta_0)$

1 **for** $i \leftarrow 0$ **to** $\rho_1 + \rho_2$
2 **do** $Q_i(x,y,z) = \min_{0\le j\le v} P_{i-j}(x,y,z)S_j(x,y,z)$
3 $\mathcal{B} = (Q_0(x,y,z), \ldots, Q_{\rho_1+\rho_2}(x,y,z))$
4 **while** $\Delta(\mathcal{B}) > \Delta_0$
5 **do** $\alpha_i \leftarrow rand(), 0 \le i \le \rho_1$
6 $\beta_j \leftarrow rand(), 0 \le j \le \rho_2$
7 $Q(x,y,z) \leftarrow (\sum_{i=0}^{\rho_1} \alpha_i S_i(x,y,z))\,(\sum_{i=0}^{\rho_2} \beta_i P_i(x,y,z))$
8 $\mathcal{B} \leftarrow$ REDUCE$(\mathcal{B}, Q(x,y,z))$
9 **return** $\mathcal{B}$

Рис. 5.9. Построение базиса Грёбнера $M_{\rho_1+\rho_2, r_1+r_2}$.

Если $\mathbb{F}$ является полем характеристики 2 и $S_i(x,y,z) = P_i(x,y,z), 0 \le i \le \rho_1 = \rho_2$, сложность этого алгоритма может быть несколько снижена путем построения $Q_{2j}(x,y,z) =$

$S_i^2(x,y,z)$ и использования выражения, представленного на строке 2 алгоритма только для нечетных i.

5.2.3.4. Двоичный интерполяционный алгоритм

Представленные выше алгоритмы могут быть использованы для построения базиса Грёбнера $M_{\rho,r}$ из базиса $M_{1,1}$ через последовательность промежуточных модулей M_{ρ_i,r_i}. Из теоремы 8 можно заключить, что для минимизации сложности следует выбирать значения ρ_i максимально возможными. В связи с этим следует поддерживать отношение ρ_i/r_i как можно более близким к ρ/r, учитывая условие целочисленности r_i и ρ_i. На рис. 5.10 представлен алгоритм, реализующий эту стратегию. В нем должно использоваться $(1,w_1,w_2)$-взвешенное лексикографическое упорядочение с $y \prec z \prec x$. Вначале строится базис Грёбнера $M_{1,1}$ (строки 1-2) на основе результата леммы 14. Алгоритм *Reduce* используется для получения линейно независимых над $\mathbb{F}[x]$ многочленов, образующих базис Грёбнера этого модуля. Далее базис этого модуля преобразуется в базис $M_{\lfloor\rho/r\rfloor,1}$, который является отправной точкой для двоичного интерполяционного алгоритма. Переменные R и Π соответствуют r_i и ρ_i.

Сложность этого алгоритма определяется последней его итерацией, поэтому ее можно оценить как $O(n^2r^3)$. Это обусловлено в основном сложностью (*Reduce*). Асимптотическая сложность алгоритма может быть снижена за счет использования обобщения алгоритма Кнута-Шёнхаге, предложенного в [26].

В качестве исходных данных для этого алгоритма должны использоваться многочлены $q_{10}(x), q_{11}(x)$, полученные в разделе 5.2.1, или многочлены $\Lambda(x), xB(x)$, вычисляемые с помощью алгоритма Берлекэмпа-Месси, как в исходном методе Ву. После завершения работы алгоритма следует выбрать наименьший многочлен $S(x,y,z)$ в полученном базисе Грёбнера относительно $(1,w_1,w_2)$-взвешенного лексикографического упорядочения и найти многочлены $a(x), b(x)$, удовлетворяющие $S(x,a(x),b(x)) = 0$. Это может быть сделано с помощью обобщенного алгоритма Рота-Рукенштейн, представленного в [193].

5.2.4. Численные результаты

Переформулированный метод Ву и предложенный двоичный интерполяционный алгоритм были реализованы программно и их сложность была изучена с помощью моделирования. Для сравнения были рассмотрены также ИИА [129] и метод Гурусвами-Судана с двоичным интерполяционным алгоритмом и методом перекодирования [165]. Заметим, что последний алгоритм требует выбора иных значений кратности корней r. Во всех случаях

```
INTERPOLATEWU(q_10(x), q_11(x), φ(x), r, ρ)
 1  G ← (zφ(x), yφ(x))
 2  G ← REDUCE(G, zq_11(x) − yq_10(x))
 3  π ← ⌊ρ/r⌋
 4  for j ← 1 to π
 5  do G ← EXPAND(G, n, 1)
 6  Π = π; B ← G; R ← 1
 7   Пусть r = Σ_{j=0}^{m} r_j 2^j, r_j ∈ {0, 1}
 8  for j ← m − 1 to 0
 9  do R ← 2R; Π ← 2Π
10      B ← MERGE(B, B, nR(R + 1)/2)
11      if r_j = 1
12        then R ← R + 1; Π ← Π + π
13             B ← MERGE(B, G, nR(R + 1)/2)
14      while ⌊Rρ/r⌋ > Π
15      do B ← EXPAND(B, n, R)
16         Π ← Π + 1
17  return B
```

Рис. 5.10. Построение базиса Грёбнера $M_{\rho,r}$

r было выбрано равным наименьшему значению, обеспечивающему исправление t ошибок. Полученные результаты представлены в таблице 5.1. Видно, что во всех случаях реализация метода By, основанная на двоичном интерполяционном алгоритме, обеспечивает в 5 и более раз лучшую производительность по сравнению с ИИА. Т.к. метод By допускает использование существенно меньшей кратности корней по сравнению с методом Гурусвами-Судана, он, в сочетании с предложенным двоичным интерполяционным алгоритмом, обеспечивает лучшую производительность даже по сравнению с наиболее эффективной реализацией алгоритма Гурусвами-Судана, использующей двоичный интерполяционный алгоритм и преобразование перекодирования [99].

Таблица 5.1. Время декодирования, с

	$(255, 219), t = 19$	$(255, 128), t = 73$	$(31, 15), t = 10$	$(63, 31), t = 19$	$(63, 20), t = 28$
By+ИИА [130]	0.56	1.1	0.33	0.073	7.88
By+двоичный	0.115	0.22	0.048	0.017	0.355
ГС+ перекодирование+ двоичный	1.94	0.83	0.11	0.065	–

5.3. Выводы

В данной главе были представлены быстрые алгоритмы для интерполяционного шага алгоритмов Гурусвами-Судана и By списочного декодирования кодов Рида-Соломона. В основе предложенных алгоритмов лежит обобщение двоичного метода возведения в степень на случай идеалов кольца многочленов, а также предложенный рандомизированный алгоритм вычисления базиса Грёбнера произведения нульмерных идеалов. Было показано, что применение предложенного подхода приводит к значительному снижению сложности списочного декодирования кодов Рида-Соломона.

Глава 6

Спектральные методы кодирования и декодирования кодов Рида-Соломона

В данной главе рассматривается задача быстрого систематического кодирования кодов Рида-Соломона и полярных кодов с ядром Рида-Соломона. Предлагаемый подход основан на циклотомическом алгоритме быстрого преобразования Фурье, введенном в магистерской диссертации автора и усовершенствованном в данной работе. Этот алгоритм основан на группировке входных или выходных элементов дискретного преобразования Фурье по циклотомическим классам, позволяющей свести исходную задачу к вычислению коротких циклических сверток.

Циклотомический алгоритм БПФ представлен в разделе 6.1. Основанный на нем быстрый алгоритм вычисления синдрома при декодировании кодов Рида-Соломона рассмотрен в разделе 6.2. Алгоритм быстрого систематического кодирования кодов Рида-Соломона представлен в разделе 6.3. Вопросы его применения в системах хранения данных и при систематическом кодировании полярных кодов с ядром Рида-Соломона освещены, соответственно, в разделах 6.4 и 6.5. Упрощенный метод поиска корней многочленов над конечным полем представлен в разделе 6.6.

6.1. Быстрое преобразование Фурье над конечным полем

При решении многих вычислительных задач, возникающих при реализации алгоритмов кодирования и декодирования, оказывается необходимым вычислять дискретное преобразование Фурье. В данном разделе описан эффективный алгоритм решения этой задачи, а также представлен пример его использования для вычисления вектора синдрома при классическом декодировании кодов Рида-Соломона.

6.1.1. Циклотомический алгоритм БПФ

Рассмотрим задачу быстрого вычисления

$$F_j = \sum_{i=0}^{n-1} f_i \alpha^{ij}, f_i \in \mathbb{F}_{2^m},$$

где $\alpha \in \mathbb{F}_{2^m}$ — примитивный корень степени $n = 2^m - 1$ из 1.

6.1.1.1. Циклотомическое разложение

Рассмотрим набор циклотомических классов по модулю n над $\mathbb{F}_2$:

$$\{0\},\{k_1,k_1 2,k_1 2^2,\ldots,k_1 2^{m_1-1}\},\ldots,\{k_l,k_l 2,k_l 2^2,\ldots,k_l 2^{m_l-1}\},$$

где $k_i \equiv k_i 2^{m_i} \bmod n$.

Многочлен $f(x)=\sum\limits_{i=0}^{n-1} f_i x^i, f_i \in \mathbb{F}_{2^m}$ может быть разложен как

$$f(x)=\sum_{i=0}^{l} L_i(x^{k_i}), \quad L_i(y)=\sum_{j=0}^{m_i-1} f_{k_i 2^j \bmod n} y^{2^j}. \tag{6.1}$$

Действительно, выражение (6.1) представляет собой способ группировки чисел $s \in [0,n-1]$ по циклотомическим классам: $s \equiv k_i 2^j \bmod n$. Очевидно, что такое разложение существует всегда. Заметим, что при $k_i=0$ свободный член f_0 мы можем записать как значение многочлена $L_0(y)=f_0 y$ при $y=x^0$.

Выражение (6.1) будем называть циклотомическим разложением многочлена $f(x)$.

Пример 12. *Многочлен* $f(x)=\sum_{i=0}^{6} f_i x^i, f_i \in \mathbb{F}_{2^3}$ *представляется как*

$$\begin{aligned} f(x) &= L_0(x^0)+L_1(x)+L_2(x^3); \\ L_0(y) &= f_0 y, \\ L_1(y) &= f_1 y+f_2 y^2+f_4 y^4, \\ L_2(y) &= f_3 y+f_6 y^2+f_5 y^4. \end{aligned}$$

6.1.1.2. Быстрое вычисление преобразования Фурье

В соответствии с разложением (6.1) запишем $f(\alpha^j)=\sum_{i=0}^{l} L_i(\alpha^{j\,k_i})$. Как известно [18], элемент α^{k_i} является корнем соответствующего минимального многочлена степени m_i и, следовательно, лежит в подполе $\mathbb{F}_{2^{m_i}}$, $m_i \mid m$. Таким образом, все величины $(\alpha^{k_i})^j$ принадлежат полю $\mathbb{F}_{2^{m_i}}$ и могут быть разложены в каком-либо базисе $(\beta_{i,0},\ldots,\beta_{i,m_i-1})$ этого поля: $\alpha^{j\,k_i}=\sum\limits_{s=0}^{m_i-1} a_{ijs}\beta_{i,s}, a_{ijs} \in \mathbb{F}_2$. Тогда значения каждого из линеаризованных многочленов могут быть вычислены в базисных точках соответствующего подполя по формуле

$$L_i(\beta_{i,s})=\sum_{p=0}^{m_i-1} \beta_{i,s}^{2^p} f_{k_i 2^p}, \quad i \in [0,l], s \in [0,m_i-1]. \tag{6.2}$$

Базисы $(\beta_{i,0},\ldots,\beta_{i,m_i-1})$ для каждого из линеаризованных многочленов $L_i(y)$ могут выбираться независимо.

В силу свойств линеаризованных многочленов, компоненты преобразования Фурье многочлена $f(x)$ являются линейными комбинациями этих значений

$$F_j = f(\alpha^j) = \sum_{i=0}^{l} \sum_{s=0}^{m_i-1} a_{ijs} L_i(\beta_{i,s}) = \sum_{i=0}^{l} \sum_{s=0}^{m_i-1} a_{ijs} \left(\sum_{p=0}^{m_i-1} \beta_{i,s}^{2^p} f_{k_i 2^p} \right), \quad j \in [0, n-1]. \tag{6.3}$$

Последнее выражение может быть записано в матричной форме как $F = ALf$, где $F = (F_0, F_1, \ldots, F_{n-1})^T$, $f = (f_0, f_{k_1}, f_{k_1 2}, f_{k_1 2^2}, \ldots, f_{k_1 2^{m_1-1}}, \ldots, f_{k_l}, f_{k_l 2}, f_{k_l 2^2}, \ldots, f_{k_l 2^{m_l-1}})^T$ есть перестановка вектора коэффициентов исходного многочлена $f(x)$, соответствующая разложению (6.1), A — матрица, составленная из элементов $a_{ijs} \in \mathbb{F}_2$, L — блочно-диагональная матрица, составленная из элементов $\beta_{i,s}^{2^p}$. Очевидно, что для линеаризованных многочленов одинаковой степени m_i, входящих в разложение (6.1), можно выбрать одинаковые базисы $(\beta_{i,s})$ в подполях $\mathbb{F}_{2^{m_i}}$, вследствие чего матрица L содержит большое число одинаковых блоков.

Таким образом, задача БПФ разбивается на два этапа: умножение блочно-диагональной матрицы L на исходный вектор f и умножение двоичной матрицы A на полученный вектор $S = Lf$

$$F = ALf. \tag{6.4}$$

Рассмотрим более подробно первый этап преобразования Фурье — задачу вычисления произведения $S = Lf$. Блочно-диагональная матрица

$$L = \begin{pmatrix} L_0 & 0 & \ldots & 0 \\ 0 & L_1 & \ldots & 0 \\ \ldots & \ldots & \ddots & \ldots \\ 0 & 0 & \ldots & L_l \end{pmatrix}$$

состоит из блоков

$$L_i = \begin{pmatrix} \beta_{i,0} & \beta_{i,0}^2 & \ldots & \beta_{i,0}^{2^{m_i-1}} \\ \beta_{i,1} & \beta_{i,1}^2 & \ldots & \beta_{i,1}^{2^{m_i-1}} \\ \ldots & \ldots & \ldots & \ldots \\ \beta_{i,m_i-1} & \beta_{i,m_i-1}^2 & \ldots & \beta_{i,m_i-1}^{2^{m_i-1}} \end{pmatrix}.$$

Выберем в качестве $(\beta_{i,0}, \ldots, \beta_{i,m_i-1})$ нормальный базис $\left(\gamma_i, \gamma_i^2, \ldots, \gamma_i^{2^{m_i-1}}\right)$. Тогда матрица L состоит из блоков вида

$$L_i = \begin{pmatrix} \gamma_i^{2^0} & \gamma_i^{2} & \dots & \gamma_i^{2^{m_i-1}} \\ \gamma_i^{2} & \gamma_i^{4} & \dots & \gamma_i^{2^0} \\ \dots & \dots & \dots & \dots \\ \gamma_i^{2^{m_i-1}} & \gamma_i^{2^0} & \dots & \gamma_i^{2^{m_i-2}} \end{pmatrix}.$$

В силу блочно-диагональной структуры матрицы L вычисление произведения $S = Lf$ может быть представлено как $S = (b_0, b_1, \dots, b_l)^T = L\,(a_0, a_1, \dots, a_l)^T$, где $b_i = (b_{i,0}, b_{i,1}, \dots, b_{i,m_i-1})$ — подвектора искомого вектора S, $a_i = (a_{i,0}, a_{i,1}, \dots, a_{i,m_i-1})$ — подвектора исходного вектора f.

Представим вычисление $b_i^T = L_i a_i^T$ как циклическую свертку

$$\begin{aligned} b_i(x) =& b_{i,0} + b_{i,m_i-1}x + \ldots + b_{i,1}x^{m_i-1} = \\ =& (\gamma_i + \gamma_i^{2^{m_i-1}}x + \ldots + \gamma_i^2 x^{m_i-1})(a_{i,0} + a_{i,1}x + \ldots + a_{i,m_i-1}x^{m_i-1}) \bmod (x^{m_i} - 1). \end{aligned}$$

Для ее вычисления могут быть применены известные алгоритмы [5, 9, 1]. При этом использование свойства нормального базиса $\gamma_i + \gamma_i^2 + \ldots + \gamma_i^{2^{m_i-1}} = 1$ позволяет заметно сократить число операций при вычислении циклической свертки. Отметим, что вычисление значений линеаризованных многочленов с помощью циклической свертки было описано в монографии [9].

Описанный подход позволяет свести задачу умножения блочно-диагональной матрицы L на исходный вектор f над $\mathbb{F}_{2^m}$ к задаче вычисления $l+1$ циклических сверток малой длины m_i. Существующие алгоритмы вычисления циклических сверток $b_i(x) = \gamma_i(x)a_i(x) \bmod (x^{m_i} - 1)$ могут быть записаны в матричном виде как

$$b_i = \begin{pmatrix} b_{i,0} \\ b_{i,1} \\ \dots \\ b_{i,m_i-1} \end{pmatrix} = Q_i \left(\left(D_i \begin{pmatrix} \gamma_i \\ \gamma_i^{2^{m_i-1}} \\ \dots \\ \gamma_i^2 \end{pmatrix} \right) \cdot (P_i a_i) \right), \tag{6.5}$$

где Q_i, D_i и P_i являются двоичными матрицами, а $x \cdot y$ обозначает покомпонентное произведение векторов. Очевидно, что вектор $C_i = D_i \left(\gamma_i, \gamma_i^{2^{m_i-1}}, \dots, \gamma_i^2\right)^T$ может быть вычислен заранее. Таким образом, выражение (6.4) может быть переписано как

$$F = AQ\,(C \cdot (Pf)), \tag{6.6}$$

где Q — двоичная блочно-диагональная матрица объединенных последующих сложений для $l+1$ циклической свертки, C — объединенный вектор констант, P — двоичная блочно-диагональная матрица объединенных предварительных сложений.

Учитывая формулы (6.4) и (6.6), второй этап БПФ может рассматриваться как умножение двоичной матрицы AQ на вектор $C \cdot (Pf)$. Для вычисления произведения $(AQ)(C \cdot (Pf))$ могут быть использованы модифицированный алгоритм "четырех русских" (В.Л.Арлазаров, Е.А.Диниц, М.А.Кронрод, И.А.Фараджев) для умножения булевых матриц [2] или алгоритм почтальона [109] со сложностью $O(n^2/\log n)$ сложений над элементами поля $\mathbb{F}_{2^m}$.

6.1.1.3. Оценка сложности алгоритма

Как было показано, задача вычисления БПФ многочлена над $\mathbb{F}_{2^m}$ сводится к набору задач вычисления циклической свертки многочленов и умножению на двоичную матрицу. Предполагая, что сложность вычисления короткой циклической свертки в конечном поле асимптотически равна $C_{conv}(t) = O(m \log^t m)$, получим, что сложность всего алгоритма равна

$$C = \sum_{i=0}^{l} C_{conv}(m_i) + C_{bm} = \sum_{i=0}^{l} O(m_i \log^t m_i) + C_{bm} =$$
$$O(\tfrac{n}{m} m \log^t m) + C_{bm} = O(n \log^t \log n) + C_{bm}.$$

Здесь C_{bm} — сложность умножения на двоичную матрицу. Асимптотическая оценка сложности процедуры умножения на двоичную матрицу, порождаемой вышеописанным алгоритмом, на момент написания данной работы автору неизвестна и требует дальнейших исследований. Если оценить сложность данного этапа через сложность алгоритма "четырех русских" $(n^2/\log n)$, то следует признать, что предложенный алгоритм БПФ эффективен только при малых значениях длины преобразования (см. таблицу 6.1), хотя известны асимптотически более эффективные алгоритмы БПФ для конечных полей со сложностью $O(n \log^2 n)$ операций в основном поле [191, 25]. Недавно был предложен алгоритм БПФ со сложностью $O(n \log_2 n)$ [111].

6.1.1.4. Сравнение сложности алгоритмов быстрого преобразования Фурье

Вычисление $a+b$ (или $a \times b$) будем считать сложением (умножением) только тогда, когда оба слагаемых (сомножителя) принадлежат основному полю [6], то есть операции в простом подполе не учитываются [5]. В таблице 6.1 приведена сложность БПФ длины $n = 2^m - 1$ над полями $\mathbb{F}_{2^m}$ в числе умножений N_{mul} и сложений N_{add}. Схема Горнера описана, например, в [9], а модификация алгоритма Герцеля для конечных полей рассмотрена в монографии [5]. Описанный метод представляет собой систематический подход к построению преобразования Фурье над конечным полем. Вместе с тем, необходимо отметить, что приведенные преобразования имеют много общего с [12]. Основные отличия состоят в следующем:

Таблица 6.1. Сложность некоторых алгоритмов БПФ

	Метод Горнера		Алгоритм Герцеля		Алгоритмы из [9] и [1]		Метод Захаровой [12]		Предлагаемый метод	
n	N_{mul}	N_{add}	N_{mul}	N_{add}	N_{mul}	N_{add}	N_{mul}	N_{add}	N_{mul}	N_{add}
7	36	42	12	42	9	35	6	26	6	25
15	196	210	38	210	20	70	16	100	16	77
31	900	930	120	930	108	645	60	388	54	315
63	3844	3906	282	3906	158	623	97	952	97	805
127	15876	16002	756	16002	594	5770	468	3737	216	2780
255	64516	64770	1718	64770	1225	4715	646	35503	586	7919
511	260100	260610	4044	260610	4374	—	—	—	1014	26643

1. Матрица L имеет регулярную структуру. Это позволяет свести задачу минимизации числа умножений к классической задаче вычисления циклической свертки.

2. Алгоритм содержит всего два умножения двоичных матриц на векторы, что может быть использовано для более глубокой оптимизации.

3. Используется более эффективный алгоритм оптимизации последовательности сложений.

6.1.2. Обратный циклотомический алгоритм

Сгруппируем номера элементов вектора компонентов ДПФ по циклотомическим классам, т.е.

$$F_{k_i 2^s} = \sum_{j=0}^{n-1} f_j \alpha^{j k_i 2^s}, 0 \le i < l. \tag{6.7}$$

Выберем нормальный базис $\{\gamma_i, \gamma_i^2, \ldots, \gamma_i^{2^{m_i-1}}\}$ подполя $\mathbb{F}_{2^{m_i}} \subset \mathbb{F}_{2^m}$, где $k_i 2^{m_i} \equiv k_i \bmod 2^m - 1$, т.е. $\alpha^{k_i} \in \mathbb{F}_{2^{m_i}}$. Тогда $\alpha^{j k_i} = \sum_{t=0}^{m_i-1} a_{itj} \gamma_i^{2^t}, a_{itj} \in \mathbb{F}_2$. Следовательно, (6.7) может быть записано как

$$F_{k_i 2^s} = f(\alpha^{k_i 2^s}) = \sum_{t=0}^{m_i-1} \gamma_i^{2^{s+t}} \sum_{j=0}^{n-1} a_{itj} f_j, \tag{6.8}$$

где $f(x) = \sum_{j=0}^{n-1} f_j x^j$. В матричном виде это выражение может быть записано как

$$F = LAf, \tag{6.9}$$

где F — переставленный вектор компонент ДПФ, L — блочно-диагональная матрица, блоки L_i которой являются циркулянтами, образованными элементами $\gamma_i^{2^s}$ нормальных базисов, и

A — двоичная матрица, образованная элементами a_{itj}. Это выражение могло быть также получено из свойств симметрии прямого и обратного ДПФ путем обращения матриц в (6.4).

Пример 13. *Рассмотрим построение алгоритма БПФ длины 7 над конечным полем* $\mathbb{F}_{2^3}$. *Пусть* α *— корень примитивного многочлена* $x^3 + x + 1$. *В качестве базиса поля* $\mathbb{F}_{2^3}$ *выберем нормальный базис* $(\gamma, \gamma^2, \gamma^4)$, *где* $\gamma = \alpha^3$. *Тогда алгоритм БПФ может быть записан в следующем виде:*

$$F = \begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{pmatrix} = ALf = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma^1 & \gamma^2 & \gamma^4 & 0 & 0 & 0 \\ 0 & \gamma^2 & \gamma^4 & \gamma^1 & 0 & 0 & 0 \\ 0 & \gamma^4 & \gamma^1 & \gamma^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \gamma^1 & \gamma^2 & \gamma^4 \\ 0 & 0 & 0 & 0 & \gamma^2 & \gamma^4 & \gamma^1 \\ 0 & 0 & 0 & 0 & \gamma^4 & \gamma^1 & \gamma^2 \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_4 \\ f_3 \\ f_6 \\ f_5 \end{pmatrix}.$$

Обратное преобразование Фурье вектора F *может быть получено следующим образом*

$$f = \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_4 \\ f_3 \\ f_6 \\ f_5 \end{pmatrix} = L^{-1}A^{-1}F = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma^1 & \gamma^2 & \gamma^4 & 0 & 0 & 0 \\ 0 & \gamma^2 & \gamma^4 & \gamma^1 & 0 & 0 & 0 \\ 0 & \gamma^4 & \gamma^1 & \gamma^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \gamma^1 & \gamma^2 & \gamma^4 \\ 0 & 0 & 0 & 0 & \gamma^2 & \gamma^4 & \gamma^1 \\ 0 & 0 & 0 & 0 & \gamma^4 & \gamma^1 & \gamma^2 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{pmatrix}.$$

Учитывая связь прямого и обратного преобразований Фурье, получим следующую эквивалентную запись алгоритма БПФ

$$\tilde{F} = \begin{pmatrix} F_0 \\ F_6 \\ F_5 \\ F_3 \\ F_4 \\ F_1 \\ F_2 \end{pmatrix} = L^{-1}A^{-1} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{pmatrix} = L^{-1}A^{-1}\tilde{f},$$

где через $\tilde{F}$ *и* $\tilde{f}$ *обозначены перестановки векторов коэффициентов* F *и* f.

Используя алгоритм вычисления трехточечной циклической свертки $b_i(x) = b_{i,0} + b_{i,2}x + b_{i,1}x^2 = (\gamma + \gamma^4 x + \gamma^2 x^2)(a_{i,0} + a_{i,1}x + a_{i,2}x^2) \bmod (x^3 - 1)$, *представленный в [5], получим*

$$b_i = \begin{pmatrix} b_{i,0} \\ b_{i,1} \\ b_{i,2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \left(\left[\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \gamma \\ \gamma^4 \\ \gamma^2 \end{pmatrix} \right] \cdot \left[\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{i,0} \\ a_{i,1} \\ a_{i,2} \end{pmatrix} \right] \right) = Q_i \left(C_i \cdot (P_i a_i) \right),$$

$i = 1, 2$. *С учетом* $\gamma + \gamma^2 + \gamma^4 = 1$ *видно, что алгоритм требует 3 умножений, 4 предварительных и 5 последующих сложений.*

Таким образом, получаем следующую запись алгоритма БПФ

$$\tilde{F} = \begin{pmatrix} F_0 \\ F_6 \\ F_5 \\ F_3 \\ F_4 \\ F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \left(\left[\begin{pmatrix} 1 \\ 1 \\ \gamma^2 + \gamma^4 \\ \gamma + \gamma^4 \\ \gamma + \gamma^2 \\ 1 \\ \gamma^2 + \gamma^4 \\ \gamma + \gamma^4 \\ \gamma + \gamma^2 \end{pmatrix} \cdot \left[\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{pmatrix} \right] \right) =$$

$$= Q(C \cdot (PA^{-1}\tilde{f})).$$

Или, перемножая матрицы P *и* A^{-1}, *имеем*

$$\tilde{F} = Q \left(\begin{pmatrix} 1 \\ 1 \\ \gamma^2 + \gamma^4 \\ \gamma + \gamma^4 \\ \gamma + \gamma^2 \\ 1 \\ \gamma^2 + \gamma^4 \\ \gamma + \gamma^4 \\ \gamma + \gamma^2 \end{pmatrix} \cdot \left[\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{pmatrix} \right] \right).$$

Таким образом, вычисление БПФ длины 7 сводится к следующей последовательности действий:

1. *выполнение предварительных сложений* $(PA^{-1}) \times \tilde{f}$:

   $V_{10} = f_3 + f_5; V_{11} = f_1 + V_{10}; V_2 = f_2 + V_{11}; V_6 = f_4 + V_{11}; V_9 = f_6 + V_2; V_{12} = f_4 + V_9; F_0 = f_0 + V_{12}; V_8 = f_5 + V_9; V_{13} = f_6 + F_0; V_1 = V_{10} + V_{13}; V_5 = V_1 + V_{12}; V_7 = V_6 + V_8; V_4 = V_7 + V_{10}; V_3 = V_2 + V_4,$

2. *выполнение умножений на константы* $C \cdot (PA^{-1}\tilde{f})$:

   $V_{14} = V_2\,\alpha; V_{15} = V_3\,\alpha^2; V_{16} = V_4\,\alpha^4; V_{17} = V_6\,\alpha; V_{18} = V_7\,\alpha^2; V_{19} = V_8\,\alpha^4;$

3. *умножение матрицы* Q *на вектор* $C \cdot (PA^{-1}\tilde{f})$*:*

$T_1 = V_1 + V_{16}; T_2 = V_{14} + V_{15}; F_6 = V_{15} + T_1; F_5 = V_1 + T_2; F_3 = V_{14} + T_1; T_3 = V_5 + V_{19}; T_4 = V_{17} + V_{18}; F_4 = V_{18} + T_3; F_1 = V_5 + T_4; F_2 = V_{17} + T_3.$

Общая сложность алгоритма составляет $2 \times 3 = 6$ *умножений и* $14 + 2 \times 5 = 24$ *сложения, что на одно сложение меньше, чем для алгоритма, представленного в табл. 6.1.*

Известно, что элементы $\alpha^{k_i 2^s}$ образуют класс сопряженности над $\mathbb{F}_{2^m}$, и их минимальный многочлен равен

$$\mu_i(x) = \prod_{s=0}^{m_i - 1} (x - \alpha^{k_i 2^s}).$$

Следовательно, (6.8) может быть переписано как

$$F_{k_i 2^s} = f(\alpha^{k_i 2^s}) = r_i(\alpha^{k_i 2^s}), \tag{6.10}$$

где $r_i(x) \equiv f(x) \bmod \mu_i(x)$. После нахождения $r_i(x)$, значения $F_{k_i 2^s}$ могут быть вычислены в соответствии с (6.8). Таким образом, ДПФ может быть вычислено как

$$F = LA'A''f,$$

где вычисление $A''f$ эквивалентно нахождению остатков от деления $f(x)$ на $\mu_i(x)$ и A' — блочно-диагональная матрица, состоящая из блоков $A'_i = (a_{itj}), 0 \le t, j < m_i$. Умножение на A'_i требует $O(m_i^2)$ операций. Однако эти операции могут быть совмещены с предварительными сложениями, требуемыми для вычисления циклических сверток, задаваемых матрицами L_i. Для вычисления $A''f$ может использоваться быстрый рекурсивный алгоритм одновременного приведения по модулю, используемый в классической процедуре одновременного вычисления значений многочлена в различных точках [187], со сложностью $2D(n)\lfloor \log l \rfloor$ операций сложения, где $D(n)$ — сложность используемого алгоритма деления. Полагая, что $D(n) = 2M(n)+n$ [187], где $M(n)$ — сложность перемножения многочленов степени n, получим, что сложность предлагаемого обратного циклотомического алгоритма равна $O(M(n)\log n)$. Вместе с тем, следует отметить, что использование быстрого алгоритма деления становится оправданным только для очень больших n.

6.2. Применение обратного преобразования Фурье для быстрого вычисления вектора синдрома

Рассмотрим далее применение описанного алгоритма БПФ для вычисления компонентов вектора синдрома при декодировании кодов Рида-Соломона. В соответствии с (1.25), они

могут быть представлены как

$$S_l = f(\alpha^l), l = 0..d-2,$$

где α — примитивный элемент поля $\mathbb{F}_{2^m}$ и $f(x) = \sum_{i=0}^{n-1} f_i x^i$ — многочлен, соответствующий принятой последовательности. Очевидно, что последовательность компонент синдрома представляет собой неполное дискретное преобразование Фурье. Воспользуемся для его вычисления алгоритмом (6.9). Т.к. в большинстве случаев $d \ll n$, где $n = 2^m - 1$ — число ненулевых элементов поля, все требуемые компоненты вектора $\tilde{F}$ покрываются небольшим числом блоков матрицы L^{-1}. Естественно, что выполнять умножение вектора $A^{-1}\tilde{f}$ на оставшиеся блоки не требуется. Это дает возможность отбросить также часть строк матрицы A^{-1}. Таким образом, получено следующее выражение для вычисления вектора синдрома (неполного ДПФ):

$$S = Q'(C' \cdot (P'f)), \tag{6.11}$$

где матрица Q' образована вычеркиванием ненужных блоков матрицы Q, вектор C' получен аналогичным образом из вектора C, матрица P' образована таким же образом из матрицы PA. Кроме того, в большинстве случаев не требуется вычислять все компоненты произведения матрицы A на отдельный блок матрицы L^{-1}, т.е. возникает задача вычисления неполной циклической свертки. Алгоритмы, являющиеся оптимальными для случая полной циклической свертки, в общем случае перестают быть таковыми в случае их применения для вычисления неполной циклической свертки. Далее будет описан подход, позволяющий преобразовать билинейный алгоритм для вычисления полной циклической свертки вида (6.5) в эффективный алгоритм для вычисления неполной циклической свертки.

Очевидно, что неполная циклическая свертка может быть вычислена путем вычеркивания части строк из матрицы Q_i в (6.5). Это может привести к образованию в ней нулевых столбцов. Ясно, что не требуется вычислять те компоненты вектора попарных произведений, которые соответствуют нулевым столбцам, что дает некоторое снижение сложности. Более того, анализируя алгоритмы вычисления циклической свертки, представленные в [5], можно заметить, что вычеркивая различные комбинации строк из матрицы Q_i, можно получить различное число нулевых столбцов в оставшейся части матрицы. Т.к. изменение порядка элементов нормального базиса (коэффициентов одного из многочленов, участвующих в циклической свертке) приводит к изменению порядка коэффициентов результирующего многочлена, можно подобрать такое упорядочение коэффициентов нормального базиса, что требуемые компоненты вектора ДПФ будут покрываться строками матрицы Q с наибольшим числом нулевых столбцов, что приведет к существенному снижению требуемого числа умножений.

Пример 14. *Рассмотрим применение алгоритма БПФ-7 из примера 13 к задаче вычисления первых двух компонент F_0 и F_1 вектора ДПФ (компоненты вектора синдрома для кода $(7,5,3)$ Рида-Соломона над $\mathbb{F}_{2^3}$. Вычеркивая избыточные строки матрицы Q, получим*

$$\begin{pmatrix} F_0 \\ F_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \left(\begin{pmatrix} 1 \\ 1 \\ \gamma^2+\gamma^4 \\ \gamma+\gamma^4 \end{pmatrix} \cdot \left[\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{pmatrix} \right] \right).$$

Это дает следующий алгоритм вычисления вектора синдрома:

$$\begin{aligned} T_0 &:= f_5 + f_6; & T_7 &:= f_3 + T_3 \\ T_1 &:= f_2 + f_4; & T_8 &:= T_1 + T_6 \\ T_2 &:= f_0 + f_3; & T_9 &:= f_5 + T_7 \\ T_3 &:= f_1 + f_4; & T_{10} &:= \alpha T_9 \\ T_4 &:= T_0 + T_2; & T_{11} &:= \alpha^2 T_5 \\ T_5 &:= T_0 + T_1; & T_{12} &:= T_{10} + T_{11} \\ T_6 &:= f_1 + T_4; & & \\ F_0 &:= T_8; & F_1 &:= T_4 + T_{12} \end{aligned}$$

Заметим, что изменение порядка элементов нормального базиса (например, $(\gamma^2, \gamma^4, \gamma)$) приведет к изменению порядка строк матрицы Q. Однако ввиду небольшой размерности рассматриваемого примера, в данном случае это не приведет к изменению числа нулевых столбцов.

Описанный метод был использован для построения алгоритма вычисления синдромного многочлена для $(255, 239, 17)$ кода Рида-Соломона (определенного, например, в стандарте IEEE 802.16a). Полученный алгоритм требует 75 умножений и 2938 сложений, в то время как применение классической схемы Горнера требует $15 \cdot 254 = 3810$ умножений и $15 \cdot 254 + 254 = 4064$ сложений. Результаты имитационного моделирования показывают, что построенный алгоритм в 8 раз быстрее аналогичного алгоритма, основанного на схеме Горнера. Точное число операций, требуемое для вычисления синдромного многочлена для различных кодов Рида-Соломона, представлено в таблице 6.2. Для сравнения там же приведена сложность аналогичных алгоритмов, полученных с помощью метода Захаровой [13, 12].

Таблица 6.2. Сложность вычисления синдрома для некоторых кодов Рида-Соломона

Код	Предложенный метод		Схема Горнера		Метод Захаровой	
(n, k, d)	N_{mul}	N_{add}	N_{mul}	N_{add}	N_{mul}	N_{add}
$(255, 253, 3)$	7	508	254	508	7	529
$(255, 251, 5)$	17	905	762	1016	18	875
$(255, 249, 7)$	27	1250	1270	1524	30	1268
$(255, 247, 9)$	37	1643	1778	2032	41	1652
$(255, 245, 11)$	45	1909	2286	2540	51	2036
$(255, 243, 13)$	55	2350	2794	3048	62	2391
$(255, 241, 15)$	65	2689	3302	3556	74	2789
$(255, 239, 17)$	75	2938	3810	4064	85	2989
$(255, 223, 33)$	149	5046	7874	8128	167	5440

6.3. Быстрое систематическое кодирование кодов Рида-Соломона

Коды Рида-Соломона применяются в системах хранения данных для защиты от отказов отдельных устройств. При этом хранимые данные кодируются некоторым (как правило, высокоскоростным) укороченным[1] (n, k) кодом Рида-Соломона, и различные символы получаемого при этом кодового слова размещаются на различных устройствах. В случае отказа устройств данные могут быть восстановлены с помощью процедуры исправления стираний. Большую часть времени устройства исправны, и производительность системы определяется сложностью используемой процедуры кодирования. В связи с этим представляет интерес разработка быстрого алгоритма кодирования кодов Рида-Соломона. Кроме того, достаточно часто в таких системах выполняется операция частичного обновления данных, которая состоит в изменении нескольких информационных символов кодового слова. При этом требуется быстро вычислить новые значения проверочных символов.

Ограничимся рассмотрением кодов над полями $\mathbb{F}_q$, где $q = 2^m$.

6.3.1. Кодирование через исправление стираний

Стандартная процедура систематического кодирования (n, k) циклического кода, в т.ч. кода Рида-Соломона, задается (1.24). Деление многочленов требует $k(n - k)$ сложения и умножений в конечном поле. Хотя существует быстрый алгоритм деления [187], он не дает

[1] Распространены также схемы кодирования, использующие (дважды) расширенные коды Рида-Соломона.

никакого выигрыша при практически значимых n and k.

Предлагаемый подход состоит в том, чтобы рассматривать проверочные символы как стертые и воспользоваться для их нахождения алгоритмом исправления стираний (например, алгоритмом Форни). Кроме того, предлагается подобрать местоположение информационных и проверочных символов в кодовом слове таким образом, чтобы соответствующие формулы допускали упрощение с учетом алгебраических свойств конечных полей. Это можно сделать, так как любое подмножество мощности k множества $\{0, \ldots, n-1\}$ образует информационную совокупность (n, k) кода Рида-Соломона.

Пусть дан код Рида-Соломона с порождающим многочленом

$$g(x) = \prod_{i=0}^{n-k-1} (x - \alpha^{b+i}).$$

Напомним, что значения стертых символов могут быть найдены с помощью

$$E_j = \underbrace{\frac{X_j^{(1-b)}}{\sum_{s=0}^{\lfloor (t-1)/2 \rfloor} \Lambda_{2s+1} X_j^{-2s}}}_{\phi_i} \Gamma(X_j^{-1}),$$

где $\Gamma(x)$ — многочлен значений стираний и $\Lambda(x)$ — многочлен локаторов стираний, X_i — локаторы стираний.

Предлагаемый алгоритм быстрого кодирования укороченного (n, k) кода Рида-Соломона над $\mathbb{F}_q, n < N = q - 1$, включает следующие шаги:

1. Подготовка: построить многочлен локаторов проверочных символов

$$\Lambda^{(c)}(x) = \sum_{i=0}^{n-k} \Lambda_i^{(c)} x^i = \prod_{i=0}^{n-k-1} (1 - \alpha^{u_i} x), \tag{6.12}$$

   где $U = \{u_0, \ldots, u_{n-k-1}\}$ — множество номеров проверочных символов в кодовом слове неукороченного кода длины N. Вычислить коэффициенты Форни

$$\phi_i^{(c)} = \frac{\alpha^{(1-b)u_i}}{\sum_{s=0}^{\lfloor (n-k-1)/2 \rfloor} \Lambda_{2s+1}^{(c)} \alpha^{-2u_i s}}. \tag{6.13}$$

2. Вычислить синдром вектора информационных символов:

$$S_i = \sum_{j=0}^{k-1} m_j \alpha^{v_j(b+i)}, 0 \le i < n - k, \tag{6.14}$$

   где $V = \{v_0, \ldots, v_{k-1}\}$ — множество номеров информационных символов в кодовом слове неукороченного кода длины $N = q - 1$.

3. Вычислить многочлен значений проверочных символов $\Gamma^{(c)}(x) \equiv S(x)\Lambda^{(c)}(x) \bmod x^{n-k}$.

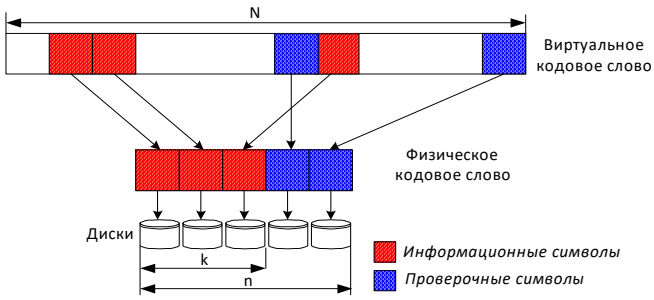


Рис. 6.1. Распределение информационных и проверочных символов в виртуальном кодовом слове

4. Вычислить проверочные символы c_{u_i} как

$$c_{u_i} = \phi_i^{(c)} \Gamma^{(c)}(\alpha^{-u_i}), 0 \le i < n-k. \tag{6.15}$$

Данный метод использует тот факт, что любые K символов $(N, K, N-K+1)$ кода МДР образуют информационную совокупность, т.е. могут быть использованы в качестве информационных. Из этих K символов k, задаваемых множеством V, действительно используются, в то время как оставшиеся $K-k$ полагаются равными нулю. Символы, не вошедшие в информационную совокупность, являются проверочными, и их номера задаются множеством U. Таким образом, имеется "виртуальное кодовое слово" длины $N = q-1$, в котором отличными от нуля могут быть только $n < N$ символов, которые отображаются на соответствующие устройства хранения данных (например, диски). Рис. 6.1 иллюстрирует этот подход.

Можно заметить, что проблема расчета значений проверочных символов сводится к вычислению значений многочлена $\sum_{j=0}^{k-1} m_j x^{v_j}$ в точках $\alpha^0, \ldots, \alpha^{n-k-1}$ (расчет синдрома), построению многочлена значений стираний $\Gamma^{(c)}(x)$ и вычислению значений $\Gamma^{(c)}(x)$ в точках α^{-u_i}. Построение практически быстрых алгоритмов решения этих задач представляется намного более простой проблемой, чем для задачи деления многочленов с остатком, которая лежит в основе классического метода систематического кодирования циклических кодов (1.24). Эти алгоритмы представлены ниже.

6.3.2. Эффективное вычисление синдрома

Как известно, компоненты синдрома вектора $(y_0, \ldots, y_{N-1})$ или соответствующего многочлена $y(x) = \sum_{i=0}^{N-1} y_i x^i$ определяются как

$$S_i = y(\alpha^{b+i}) = \sum_{j=0}^{N-1} y_j \alpha^{(b+i)j}, 0 \le i < n-k. \tag{6.16}$$

Заметим, что

$$y(x) = \underbrace{\sum_{i=1}^{k} m_i x^{v_i}}_{y'(x)} + \underbrace{\sum_{i=1}^{n-k} c_{u_i} x^{u_i}}_{y''(x)}, \tag{6.17}$$

где $y'(x)$ соответствует информационным символам, а $y''(x)$ — проверочным. На этапе кодирования можно считать, что $y''(x) = 0$.

6.3.2.1. Вычисление синдрома информационных символов

Для вычисления синдрома информационных символов может быть использован метод, описанный в разделе 6.2. Т.е. вектор синдрома может быть представлен как

$$S = \overline{L}\,\overline{A}y, \tag{6.18}$$

где $\overline{L}$ получается вычеркиванием ненужных строк из блочно-диагональной матрицы L, а также образующихся при этом нулевых столбцов. Матрица $\overline{A}$ образована строками A, соответствующими оставшимся ненулевым столбцам L. Используя быстрые алгоритмы циклической свертки для умножения на блоки L, получим следующий быстрый алгоритм вычисления синдрома:

$$S = P\left(d \cdot \underbrace{R\overline{A}}_{V} y\right). \tag{6.19}$$

Компьютерная оптимизация может быть использована для построения эффективной процедуры вычисления Vy.

Напомним, что при кодировании стертые проверочные символы y полагаются равными нулю. Это позволяет исключить соответствующие столбцы из матрицы V, оставив только те, которые соответствуют коэффициентам $y'(x)$ (см. (6.17)).

Пример 15. *Рассмотрим задачу вычисления синдрома информационного вектора m при кодировании $(10, 7, 3)$ кода Рида-Соломона. Будем считать, что информационные символы расположены в позициях $V = \{0, 1, 2, 3, 4, 5, 6\}$ в виртуальном кодовом слове. Обратный циклотомический алгоритм БПФ имеет вид*

$$\begin{pmatrix} S_0 \\ S_1 \\ S_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma_8 & \gamma_8^2 & \gamma_8^4 & \gamma_8^8 & \gamma_8^{16} & \gamma_8^{32} & \gamma_8^{64} & \gamma_8^{128} \\ 0 & \gamma_8^2 & \gamma_8^4 & \gamma_8^8 & \gamma_8^{16} & \gamma_8^{32} & \gamma_8^{64} & \gamma_8^{128} & \gamma_8 \end{pmatrix} \underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{A} \begin{pmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \end{pmatrix},$$

где $\gamma_8 = \alpha^5$. *Таким образом,* S_1 *и* S_2 *могут быть получены как коэффициенты* $b_{10}, b_{1,7}$ *многочлена* $b_1(x)$, *где* $b_1(x)x^l = (\gamma_8+\gamma_8^{128}x+\gamma_8^{64}x^2+\gamma_8^{32}x^3+\gamma_8^{16}x^4+\gamma_8^8x^5+\gamma_8^4x^6+\gamma_8^{128}x^7)a_1(x)x^l \bmod (x^8-1)$ *и коэффициенты* $a_1(x)$ *задаются произведением последних* 8 *строк* A *и т. Изучая матрицу постсложений быстрого алгоритма 8-точечной циклической свертки, приведенного в [41], можно увидеть, что оптимальное значение* l *равно* 7, *т.к. в этом случае можно исключить* 12 *столбцов из матрицы постсложений. Таким образом, получим*

$$\begin{pmatrix} S_0 \\ S_1 \\ S_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \cdot \left(\begin{pmatrix} 1 \\ \gamma_8^{128} \\ \gamma_8^{128}+\gamma_8 \\ \gamma_8^{128}+\gamma_8^{64} \\ \gamma_8^8+\gamma_8^2 \\ \gamma_8^{16}+\gamma_8^8+\gamma_8^4+\gamma_8^2 \\ \gamma_8^8+\gamma_8^4+\gamma_8^2+\gamma_8 \\ \gamma_8^{32}+\gamma_8^2 \\ \gamma_8^{64}+\gamma_8^{32}+\gamma_8^4+\gamma_8^2 \\ \gamma_8^{32}+\gamma_8^{16}+\gamma_8^2+\gamma_8^1 \\ \gamma_8^{128}+\gamma_8^{32}+\gamma_8^8+\gamma_8^2 \\ 1 \\ 1 \end{pmatrix} \cdot \left(\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} A \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix} \right) \right)$$

Этот подход может быть использован для построения быстрого алгоритма вычисления синдрома для любой комбинации стертых символов. Это потребовало бы построения набора быстрых алгоритмов умножения подматриц, образованных всевозможными комбинациями столбцов матрицы A, и соответствующих подвекторов y. Даже в случае задачи кодирования, в контексте систем хранения данных такой подход является крайне непрактичным, т.к. достаточно часто возникает необходимость выборочного обновления информационных символов кодового слова, поэтому число таких комбинаций оказывается слишком большим.

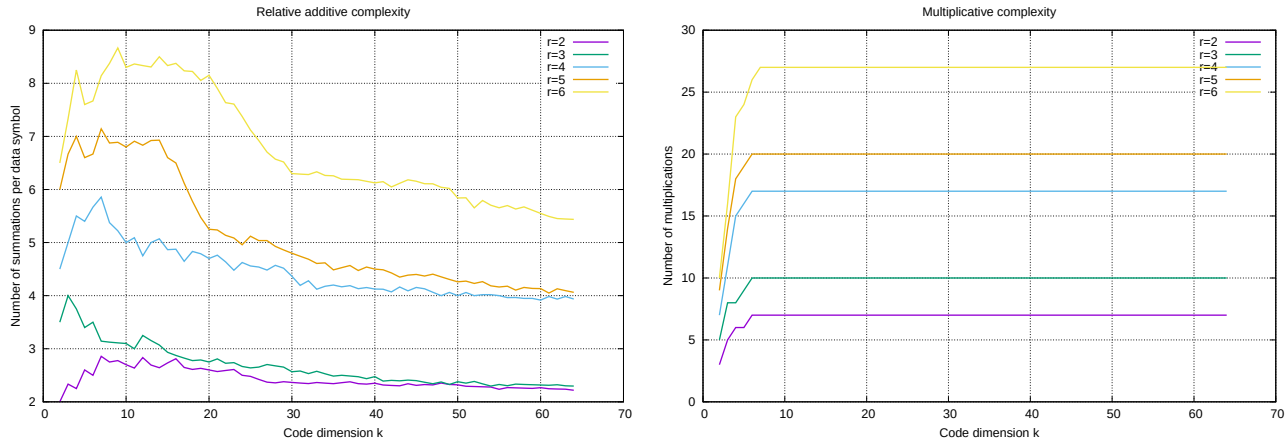
С целью минимизации сложности реализации (см. раздел 6.3.1), предлагается поместить информационные символы на k последовательных позиций в начале виртуального кодового слова, за исключением тех, которые используются для размещения проверочных символов. Если необходимо вычислить синдром информационного подвектора $(m_v, m_{v+1}, \ldots, m_{v+\delta})$, получим

$$S_i = \sum_{j=v}^{v+\delta} m_j \alpha^{ij} = \alpha^{vi} \underbrace{\sum_{j=0}^{\delta} m_{v+j} \alpha^{ij}}_{\tilde{S}_i}. \tag{6.20}$$

Значения $\tilde{S}_i$ могут быть вычислены с помощью вышеописанного обратного циклотомического алгоритма. Если оптимизированные алгоритмы для расчета (6.19) доступны только для векторов y длины μ, можно воспользоваться разложением

$$S_i = \alpha^{vi} \sum_{J=0}^{\delta/\mu-1} \alpha^{\mu J i} \underbrace{\sum_{j=0}^{\mu-1} m_{J\mu+v+j} \alpha^{ij}}_{\tilde{S}_i^{(J)}}, 0 \le i < r = n-k \tag{6.21}$$

где $(\tilde{S}_0^{(J)}, \ldots, \tilde{S}_{n-k-1}^{(J)})$ могут быть вычислены с помощью оптимизированных алгоритмов. Это позволяет существенно сократить объем программного кода, реализующего предлагаемый



(*а*) Нормированная аддитивная сложность (*б*) Мультипликативная сложность

Рис. 6.2. Сложность алгоритмов вычисления синдрома

подход. Хотя этот метод и требует большего числа умножений по сравнению с непосредственным применением (6.19), он позволяет снизить число кэш-промахов при программной реализации, т.к. доступ к смежным элементам вектора m происходит в близкие моменты времени.

Оптимизированные алгоритмы, реализующие обратный циклотомический алгоритм вычисления синдрома были сгенерированы для $\mu \leq 65$. На рис. 6.2 представлена зависимость число сложений A и умножений M в этих алгоритмах от размерности кода k для различных значений числа проверочных символов $r = n - k$. Можно заметить, что мультипликативная сложность этих алгоритмов не зависит от размерности при $k \geq 8$, а число сложений при фиксированном $r > 2$ и достаточно большом k составляет менее r операций на информационный символ.

Сложность этапа вычисления синдрома в предлагаемом методе систематического кодирования составляет $O((n-k)\log^t \log n), t \leq 2$, умножений. При использовании асимптотически быстрого варианта обратного циклотомического алгоритма БПФ требуемое число сложения равно $O(M(n)\log n)$.

6.3.2.2. Вычисление синдрома проверочных символов

Синдром необходимо вычислять не только при кодировании, но и при исправлении стираний, возникающих при отказе устройств в системе хранения данных. В разделе 6.3.3 представлен метод выбора позиций проверочных символов, обеспечивающий существенное снижение сложности кодирования. Применение этого метода приводит к рассеиванию проверочных символов по виртуальному кодовому слову, что делает многочлен $y(x)$ (см. (6.16)) разрежен-

ным. Очевидно, что применение алгоритма БПФ, разработанного для плотных многочленов, к разреженным является весьма неэффективным подходом. В связи с этим в данном разделе предлагается метод преобразования многочлена проверочных символов $y''(x)$ (см. (6.17)) к виду, допускающему эффективное применение метода, описанного в разделе 6.3.2.1.

Заметим, что для кода Рида-Соломона над $\mathbb{F}_{2^m}$ можно построить его двоичный подкод БЧХ с порождающим многочленом $\tilde{g}(x) \in \mathbb{F}_2[x]$, который делится на порождающий многочлен кода Рида-Соломона. Следовательно, $y''(\alpha^{b+i}) = \tilde{r}(\alpha^{b+i}), 0 \le i < n-k$, где $\tilde{r}(x) \equiv y''(x) \bmod \tilde{g}(x)$. Таким образом, можно привести $y''(x)$ по модулю $\tilde{g}(x)$ это требует только сложений), получив многочлен степени не выше $\deg \tilde{g}(x) - 1$, после чего воспользоваться вышеописанным методом для вычисления синдрома $\tilde{r}(x)$, который является плотным многочленом малой степени. Приведение по модулю $\tilde{g}(x)$ эквивалентно умножению вектора коэффициентов исходного многочлена на некоторую двоичную матрицу. Это может быть выполнено посредством алгоритма “четырех русских” или почтальона.

Пример 16. *Рассмотрим код Рида-Соломона над $\mathbb{F}_{2^8}$ с порождающим многочленом $g(x) = (x+\alpha^0)(x+\alpha^1)$ и проверочными символами, размещенными в позициях $V = \{85, 170\}$. Можно заметить, что*

$$\tilde{g}(x) = LCM((x+\alpha^0), M^{(\alpha)}(x)) = (x+1)(x^8+x^4+x^3+x^2+1) = x^9+x^8+x^5+x^2+x+1,$$

и

$$\begin{aligned}\tilde{r}(x) &= c_{170}x^8 + (c_{85}+c_{170})x^7 + (c_{85}+c_{170})x^6 + c_{85}x^4 + c_{170}x^3 + c_{85}x^2 + (c_{85}+c_{170})x^1 \\ &\equiv c_{85}x^{85} + c_{170}x^{170} \bmod \tilde{g}(x),\end{aligned}$$

т.е.

$$(\tilde{r}_0, \ldots, \tilde{r}_8) = \begin{pmatrix} c_{85} & c_{170} \end{pmatrix} \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Вычисление этого выражения требует лишь одного сложения ($c_{85} + c_{170}$).

Заметим, что

$$S_i = y(\alpha^{b+i}) = y'(\alpha^{b+i}) + y''(\alpha^{b+i}) = (y'(x) + \tilde{r}(x))|_{x=\alpha^{b+i}},$$

т.е. можно вычислить $z(x) = \tilde{r}(x) + y'(x)$ и применить алгоритм, описанный в п. 6.3.2.1 к $z(x)$.

6.3.3. Выбор проверочных символов

Сложность расчета (6.15), т.е. вычисления значений $\Gamma^{(c)}(x)$ в точках α^{-u_i}, также может быть снижена путем использования инструментария БПФ. Реализация этой операции с

помощью схемы Горнера потребовала бы $r(r-1)$ сложений и умножений, где $r = n-k$. Простейшим способом упростить вычисления является выбор U в качестве объединения нескольких циклотомических классов[2]. При этом множество $\{-u_i \bmod N | u_i \in U\}$ также может быть представлено в виде объединения того же числа циклотомических классов. Тогда можно воспользоваться обратным циклотомическим алгоритмом, оставив только первые $n-k$ столбцов матрицы A и строки, соответствующие выбранным циклотомическим классам.

Пример 17. *Рассмотрим задачу быстрого кодирования кода Рида-Соломона с* $r = n-k = 3$ *проверочными символами над* $\mathbb{F}_{2^8}$. *Пусть* $U = \{0, 85, 170\}$. *Нормальный базис подполя* $\mathbb{F}_2 \subset \mathbb{F}_{2^8}$ *состоит из элемента* $\gamma_1 = 1$, *в то время как для подполя* $\mathbb{F}_{2^2} \subset \mathbb{F}_{2^8}$ *он порождается* $\gamma_2 = \alpha^{85}$, *где* α *— примитивный элемент* $\mathbb{F}_{2^8}$. *Таким образом, значения* $\Gamma^{(c)}(x) = \Gamma_0^{(c)} + \Gamma_1^{(c)} x + \Gamma_2^{(c)} x^2$ *в точках* $\alpha^{-u_i}, u_i \in U$, *т.е.*

$$\Gamma^{(c)}(\alpha^0) = \Gamma_0^{(c)} + \Gamma_1^{(c)} + \Gamma_2^{(c)} \tag{6.22}$$

$$\Gamma^{(c)}(\alpha^{-85}) = \Gamma_0^{(c)} + \Gamma_1^{(c)} \alpha^{170} + \Gamma_2^{(c)} \alpha^{85} \tag{6.23}$$

$$\Gamma^{(c)}(\alpha^{-170}) = \Gamma_0^{(c)} + \Gamma_1^{(c)} \alpha^{85} + \Gamma_2^{(c)} \alpha^{170}, \tag{6.24}$$

могут быть вычислены как

$$\begin{pmatrix} \Gamma^{(c)}(\alpha^0) \\ \Gamma^{(c)}(\alpha^{-85}) \\ \Gamma^{(c)}(\alpha^{-170}) \end{pmatrix} = \begin{pmatrix} \gamma_1 & 0 & 0 \\ 0 & \gamma_2 & \gamma_2^2 \\ 0 & \gamma_2^2 & \gamma_2 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} \Gamma_0^{(c)} \\ \Gamma_1^{(c)} \\ \Gamma_2^{(c)} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \left(\begin{pmatrix} 1 \\ \gamma_2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \Gamma_0^{(c)} \\ \Gamma_1^{(c)} \\ \Gamma_2^{(c)} \end{pmatrix} \right).$$

Здесь был использован быстрый алгоритм 2-точечной циклической свертки и тождество $\gamma_2 + \gamma_2^2 = 1$. *Таким образом, значения* $\Gamma^{(c)}(x)$ *в указанных точках могут быть вычислены с помощью всего лишь 6 сложений (т.е. операций "исключающее ИЛИ") и одного умножения на* γ_2. *По сравнению с непосредственной реализацией, задаваемой* (6.22)–(6.24), *данный подход требует на 3 умножения меньше.*

Еще одним преимуществом предложенного метода назначения проверочных символов является то, что он приводит к многочлену $\Lambda^{(c)}(x)$ с двоичными коэффициентами. Действительно, из $\Lambda^{(c)}(\alpha^{-u_i}) = 0$ вытекает, что $(\Lambda^{(c)}(\alpha^{-u_i}))^2 = \tilde{\Lambda}^{(c)}(\alpha^{-2u_i}) = 0$, где $\Lambda^{(c)}(x) = \sum_{i=0}^{n-k} \Lambda_i^{(c)} x^i$ и $\tilde{\Lambda}^{(c)}(x) = \sum_{i=0}^{n-k} (\Lambda_i^{(c)})^2 x^i$. Однако предлагаемый подход предполагает, что $2u_i \in U$, т.е. $\Lambda^{(c)}(\alpha^{-2u_i}) = 0$. Следовательно, множества корней и степени $\Lambda^{(c)}(x)$ и $\tilde{\Lambda}^{(c)}(x)$ совпадают. Из этого следует, что $\Lambda^{(c)}(x) = \tilde{\Lambda}^{(c)}(x)$, т.е. $(\Lambda_i^{(c)})^2 = \Lambda_i^{(c)}$. Таким образом, $\Lambda_i^{(c)} \in \mathbb{F}_2 = \{0, 1\}$.

[2] Такое представление возможно для любых $n-k$ при m, равном степени 2.

Это позволяет полностью избежать умножений при расчете $\Gamma^{(c)}(x)$ через (1.26). В некоторых случаях оказывается, что $\Lambda^{(c)}(x) = x^{n-k}+1 \equiv 1 \bmod x^{n-k}$, т.е. многочлен значений ошибок получается как $\Gamma^{(c)}(x) = S(x)$. Для этого необходимо и достаточно, чтобы $n-k$ делило $q-1$ и выбрать $u_i = \frac{q-1}{n-k} i, 0 \le i < n-k,$.

Пример 18. *При* $U = \{0, 85, 170\}$ *и* $q = 256$ *получим* $\Lambda_c(x) = x^3 + 1 \equiv 1 \bmod x^3$.

Если $\Lambda^{(c)}(x) = x^{n-k} + 1$, то из (6.13) вытекает, что $\phi_i^{(c)} = \frac{\alpha^{(1-b)j_i}}{\alpha^{-(n-k-1)j_i}} = \frac{\alpha^{-b\frac{q-1}{n-k}i}}{\alpha^{-(q-1)i}} = \alpha^{-b\frac{q-1}{n-k}i}$. Полагая $b = 0$, получим $\phi_i^{(c)} = 1$. Таким образом, можно исключить $n-k$ умножений.

Даже если нет возможности получить $\Lambda^{(c)}(x) = x^{n-k} + 1$, можно выбрать U так, чтобы число ненулевых членов в $\Lambda^{(c)}(x)$ было бы минимальным, что позволяет снизить число сложений при вычислении $\Gamma^{(c)}(x)$. В таблице 6.3 приведены оптимизированные множества локаторов проверочных символов U, а также сложность вычисления $\Gamma(\alpha^{-u_i}), u_i \in U$, для различных значений числа проверочных символов $r = n-k$. Можно заметить, что предлагаемый подход обеспечивает значительное снижение сложности по сравнению с непосредственной реализацией (т.е. схемой Горнера). Следует также отметить, что помимо снижения сложности при расчете $\Gamma^{(c)}(\alpha^{-u_i})$, данный подход позволяет полностью исключить умножения на этапе вычисления $\Gamma^{(c)}(x) \equiv S(x)\Lambda^{(c)}(x) \bmod x^{n-k}$. Этот подход, однако, не может быть использован при исправлении произвольных стираний (например, при восстановлении данных после отказа части дисков).

Заметим, что вычисление $\Gamma^{(c)}(x)$ эквивалентно умножению $(S_0, \ldots, S_{n-k-1})$ на двоичную матрицу, задаваемую коэффициентами $\Lambda^{(c)}(x)$. Первый этап обратного циклотомического алгоритма БПФ также состоит в умножении на двоичную матрицу. Эти операции могут быть объединены, и построена единая оптимизированная процедура вычисления значений $\Gamma^{(c)}(x)$ по $(S_0, \ldots, S_{n-k-1})$.

Число умножений, требуемое при вычислении значений проверочных символов, равно $O((n-k)\log^t \log n)$. При использовании асимптотически быстрого варианта обратного циклотомического алгоритма БПФ число сложений равно $O(M(n-k)\log(n-k))$.

6.4. Применение кодов Рида-Соломона в системах хранения данных

6.4.1. Компромисс между скоростью кодирования и восстановления

Выше было показано, что при $(n-k)|(q-1)$ надлежащий выбор локаторов проверочных символов позволяет полностью избежать умножений на $\phi_i^{(c)}$ на этапе кодирования. Это невозможно сделать при $(n-k) \nmid (q-1)$. Но, т.к. большую часть времени система хранения

Таблица 6.3. Оптимизированные локаторы проверочных символов, $q = 2^8$.

r	U	$\Lambda_c(x)$	Число сложений	Число умножений
1	{0}	$x+1$	0	0
2	{85,170}	x^2+x+1	3	1
3	{0,85,170}	x^3+1	6	1
4	{17,34,68,136}	x^4+x^3+1	15	5
5	{0,51,102,153,204}	x^5+1	16	5
6	{170,85, 17,34,68,136}	$x^6+x^3+x^2+x+1$	22	6
7	{0,85,170, 17,34,68,136}	$x^7+x^6+x^4+1$	27	6
8	{119,238, 221, 187, 51, 102, 204, 153}	$x^8+x^7+x^6+x^4+1$	40	10
9	{119,238, 221, 187,51, 102, 204, 153, 0}	$x^9+x^6+x^5+x^4+x+1$	43	10
10	{85, 170, 119, 238, 221, 187, 51, 102, 204, 153}	$x^{10}+x^8+x^5+x^4+x^2+x+1$	53	11
11	{0, 85, 170, 119, 238, 221, 187, 51, 102, 204, 153}	$x^{11}+x^{10}+x^9+x^8+x^6+x^4+x^3+1$	55	11
12	{119, 238, 221, 187, 51, 102, 204, 153, 17, 34, 68, 136}	$x^{12}+x^9+x^6+x^3+1$	74	15
13	{0, 119, 238, 221, 187, 51, 102, 204, 153, 17, 34, 68, 136}	$x^{13}+x^{12}+x^{10}+x^9+x^7+x^6+x^4+x^3+x+1$	81	15
14	{85, 170, 119, 238, 221, 187, 51, 102, 204, 153, 17, 34, 68, 136}	$x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^9+x^8+x^7+x^6+x^5+x^4+x^3+x^2+x+1$	91	16
15	{0, 85,170, 119, 238, 221, 187, 51, 102, 204, 153, 17, 34, 68, 136}	$x^{15}+1$	96	16
16	{119, 238, 221, 187, 51, 102, 204, 153, 95, 190, 125, 250, 245, 235, 215, 175}	$x^{16}+x^{14}+x^{13}+x^{11}+x^7+x+1$	143	28
17	{0,15, 30, 60, 120, 240, 225,195,135, 45,90, 180, 105, 210, 165, 75, 150}	$x^{17}+1$	177	28

пребывает в исправном состоянии, а процедура восстановления задействуется только при отказе некоторых дисков, можно отложить умножение на $\phi_i^{(c)}$ до момента декодирования (т.е. исправления стираний, возникших из-за отказа устройств). Фактически это приводит к обобщенному коду Рида-Соломона (см. определение 4) с $\omega_{v_i} = 1, \omega_{u_j} = 1/\phi_j^{(c)}, 1 \le i \le 1, 1 \le j \le n-k$. Очевидно, это никак не влияет на корректирующую способность кода. Этот прием позволяет снизить сложность кодирования на $n-k$ умножений, но увеличивает сложность исправления стираний на эту же величину. Но т.к. кодирование, как правило, является более простой операцией по сравнению с исправлением стираний, а последняя операция выполняется относительно редко, относительное снижение сложности кодирования оказывается большим, чем относительное увеличение сложности при исправлении стираний.

6.4.2. Частичное обновление

В некоторых случаях может потребоваться изменить значения отдельных информационных символов кодового слова, а также соответствующих проверочных символов. Эта операция может производиться двумя способами:

- Чтение старых значений обновляемых информационных символов и проверочных символов, вычисление разности значений новых и старых информационных символов, кодирование полученного вектора разности, внесение поправки в проверочные символы.
- Чтение информационного вектора, замена в нем значений некоторых информационных символов, кодирование обновленного информационного вектора.

Первый способ целесообразно использовать при обновлении небольшого числа информационных символов, когда сложность вычисления разности и внесения поправки мала по сравнению со сложностью кодирования всего информационного вектора, а также затратами на чтение старых значений обновляемых информационных и проверочных символов.

6.4.3. Сводное описание предлагаемого подхода

В данном разделе приведено подробное описание предлагаемого метода кодирования для систем хранения данных, использующих коды Рида-Соломона над $\mathbb{F}_{2^{2^\mu}}$.

1. Проверочные символы назначаются на позиции, задаваемые множеством U, которое представлет собой объединение нескольких циклотомических классов по модулю $2^{2^\mu}-1$. Информационные символы назначаются на позиции из множества V, которое содержит k наименьших неотрицательных целых чисел, не входящих в U.

2. Систематическое кодирование выполняется путем расчета синдрома $S(x)$ информационных символов, преобразования его в многочлен значений проверочных символов $\Gamma^{(c)}(x)$ и расчета его значений в помощью (6.15).

3. Коэффициенты $\phi_i^{(c)}$ могут быть опущены в (6.15). Данный прием будет далее называться отложенным умножением Форни.

4. Частичное обновление данных осуществляется путем систематического кодирования вектора разности обновляемых информационных символов, или путем полного систематического кодирования обновленного вектора.

5. В случае отказов дисков стертые символы восстанавливаются путем построения[3] многочлена локаторов стираний $\Lambda(x)$, нахождения синдромного многочлена[4] $S(x)$, построения многочлена значений стираний $\Gamma(x)$ в соответствии с (1.26). Собственно значения стертых символов находятся из (1.5.2).

6. Во всех случаях коэффициенты синдромного многочлена $S(x)$ вычисляются с помощью сгенерированных линейных программ частичного БПФ. Если размер исходных данных (т.е. размерность кода) превышает максимальное значение, поддерживаемое этими программами, осуществляется разбиение данных на блоки меньшего размера в соответствии с (6.21).

Шаги 1, 3 ранее в литературе описаны не были.

6.4.4. Численные результаты

На рис. 6.3 представлена относительная аддитивная сложность (т.е. число A операций "исключающее ИЛИ", поделенное на k'), где $k'-1$ — степень многочлена, соответствующего информационному вектору[5] полученных процедур вычисления синдрома. Видно, что эффективность предложенного подхода растет с увеличением k'. Число умножений M практически не зависит от k', как видно из рис. 6.4.

В таблице 6.4 представлена сложность[6] систематического кодирования для различных конфигураций систем хранения данных. Здесь $k+r$ в первом столбце обозначает код Рида-

[3] Эта операция должна быть выполнена однократно при обнаружении отказа дисков.

[4] Если $\phi_i^{(c)}$ были опущены при кодировании, проверочные символы должны быть домножены на эти значения перед вычислением синдрома.

[5] Напомним, что k' может быть больше размерности кода из-за необходимости размещения проверочных символов в избранных позициях кодового слова.

[6] С учетом отложенного умножения Форни

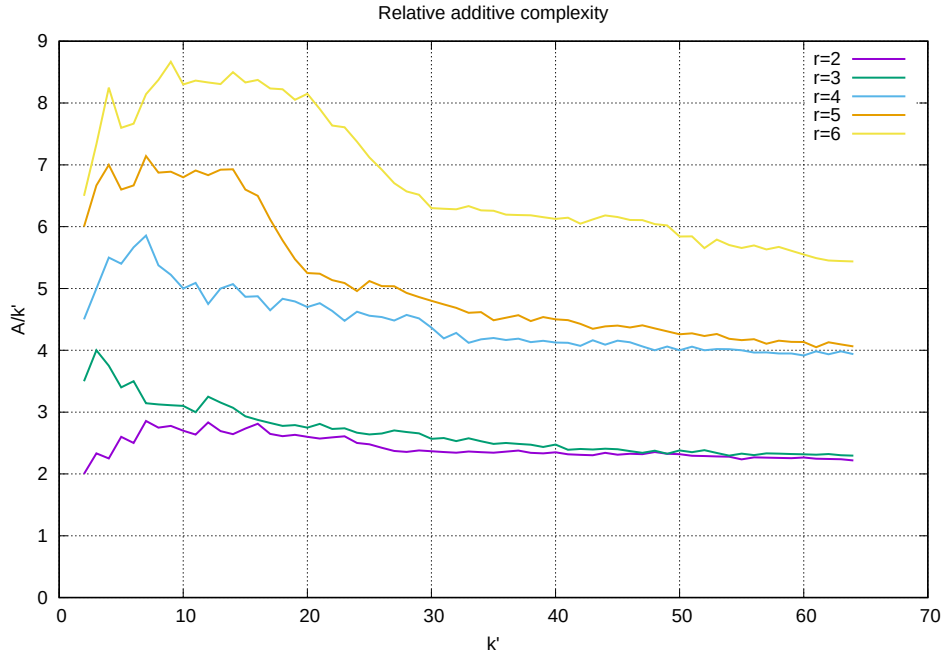


Рис. 6.3. Нормированная аддитивная сложность вычисления синдрома.

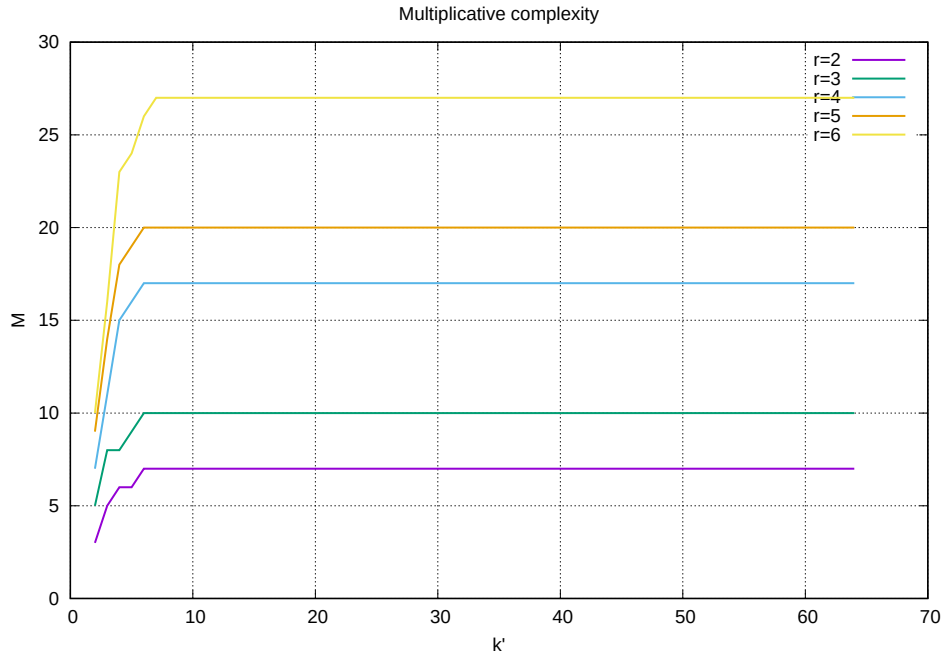


Рис. 6.4. Мультипликативная сложность вычисления синдрома.

Таблица 6.4. Сложность систематического кодирования кодов Рида-Соломона

Конфигурация	Число XOR		Число умножений в $\mathbb{F}_{2^8}$	
	всего	на символ	всего	на символ
4+2	11	2.75	7	1.75
16+3	54	3.375	11	0.6875
32+4	151	4.71875	22	0.6875
48+5	223	4.6458	25	0.52083
62+6	366	5.903	33	0.532
92+7	617	6.703	72	0.779
126+8	990	7.856	91	0.71
148+9	1214	8.202	140	0.94
168+10	1601	9.529	164	0.975
192+11	1906	9.92	230	1.19
215+12	2533	11.78	268	1.26
238+13	2827	12.75	283	1.18
241+14	3229	13.4	315	1.30
240+15	3316	13.816	330	1.375
239+16	3753	15.53	373	1.56
238+17	3764	15.81	392	1.645

Соломона размерности (т.е. число дисков, хранящих полезные данные) k с r проверочными символами. Можно заметить, что для рассмотренных случаев кодирование требует примерно 1 умножения и менее r сложений (кроме случая $r < 5$, где оно близко к r) на один информационный символ. Отметим, что стандартный алгоритм систематического кодирования кодов Рида-Соломона требует r умножений и r сложений на один информационный символ, и операция умножения в поле Галуа является значительно более трудоемкой, чем сложение. Таким образом, предлагаемый подход обеспечивает значительное снижение сложности.

В случае исправления t произвольных стираний расчет $\Gamma(x)$ требует $t(t+1)/2$ умножений и $t(t-1)/2$ сложений. Стертые символы могут быть найдены из (1.5.2) с помощью t^2 умножений и $t(t-1)$ сложений. При использовании отложенного умножения Форни необходимо выполнить $n-k$ дополнительных умножений перед вычислением синдрома.

В таблице 6.5 представлено сравнение производительности программной реализации предложенного метода и библиотеки Jerasure [138, 139], широко используемой в программно определяемых СХД. Кодирование данных осуществлялось блоками размером λk байт,

Таблица 6.5. Производительность библиотек Jerasure и программной реализации предложенного метода кодирования, ГБ/с

k	$r = n - k$	Jerasure 2.0	Предложенный метод
9	3	4.7	10.3
16	3	4.9	14.1
30	5	2.8	10.0
10	6	2.3	3.3
10	8	1.7	2.2
20	11	1.2	2.5

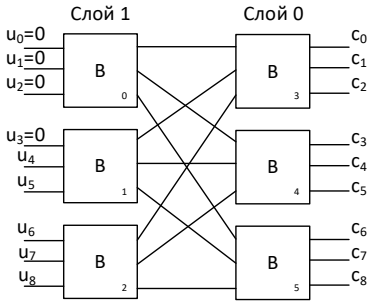


Рис. 6.5. Несистематический кодер полярного кода

где $\lambda = 4096$. Видно, что использование предложено метода систематического кодирования обеспечивает повышение производительности до 3.6 раз, причем наиболее значительный выигрыш достигается при больших k.

6.5. Быстрое систематическое кодирование полярных кодов с ядром Рида-Соломона

В данном разделе представлено обобщение алгоритма систематического кодирования полярных кодов с ядром Арикана [31] на случай кодов с ядром Рида-Соломона.

6.5.1. Декодирование полярных кодов с исправлением стираний

Рассмотрим алгоритм исправления стираний в полярном коде с ядром Рида-Соломона, пример кодера которого представлен на рис. 6.5. Представленный алгоритм использует представление кода в виде обобщенного каскадного и может рассматриваться как реализация многошагового алгоритма декодирования.

6.5.1.1. Итеративный алгоритм исправления стираний

Рассмотрим $l \times N$ матрицу C, соответствующую кодовому слову ОКК с внутренними кодами Рида-Соломона. Заметим, что столбцы матриц X и C связаны соотношением $C_{-,i} = F_l^T X_{-,i}, 0 \le i < N$, т.е.

$$C_{j,i} = \sum_{t=0}^{l-1} (F_l)_{tj} X_{t,i} = \sum_{t=0}^{l-1} \alpha_j^{l-1-t} X_{t,i}. \tag{6.25}$$

Это выражение аналогично определению 3 кодов Рида-Соломона.

Лемма 19. *Рассмотрим ОКК с внутренними* $(l, l-j, j+1), 0 \le j < l$, *кодами Рида-Соломона. Предположим, что произошли* $s \le l$ *стираний в* i*-ом столбце матричного представления* C *кодового слова. Если значния* $X_{t,i}, 0 \le t < s$ *известны, то эти стирания могут быть исправлены.*

Доказательство. Пусть E — множество таких номеров t, что $C_{t,i}$ стерт. Выражение (6.25) можно переписать как

$$C_{j,i} = \underbrace{\sum_{t=0}^{l-s-1} \alpha_j^t X_{l-1-t,i}}_{\tilde{c}_j} + \underbrace{\sum_{t=l-s}^{l-1} \alpha_j^t X_{l-1-t,i}}_{\widehat{c}_j}.$$

Первый член может рассматриваться как j-ый символ $(l, l-s, s+1)$ кода Рида-Соломона, в то время как второй член зависит исключительно от $X_{t,i}, 0 \le t < s$. Применяя алгоритм исправления стираний к вектору $(C_{0,i} - \widehat{c}_0, \ldots, C_{l-1,i} - \widehat{c}_{l-1})$, где значения в позициях $j \in E$ заменены на символ стирания ϵ, можно найти $\tilde{c}_j$ и $C_{j,i}$. □

Эта лемма позволяет использовать для исправления стираний многошаговый алгоритм декодирования, представленный на рис. 1.1, *б*, для декодирования $(Nl, \sum_{i=0}^{l-1} K_i, \ge \min_i D_i(i+1))$ ОКК с внутренними кодами Рида-Соломона длины l. Аналогичный подход был предложен в [136].

Для доказательства корректности этого алгоритма необходимо убедиться в том, что его шаг 2 всегда может быть выполнен.

Лемма 20. *После того, как многошаговый алгоритм достиг* j*-ой итерации, каждый столбец* C *имеет* 0 *или не менее* $j+1$ *стираний.*

Доказательство. Данное утверждение является тривиальным при $j = 0$. Предположим, что оно справедливо для некоторого $j < j' < l-1$, и рассмотрим действия, выполняемые на итерации j. Если декодирование внешнего кода на шаге 3 успешно, то значения $X_{t,i}, 0 \le t \le j$ доступны, поэтому лемма 19 позволяет исправить $j+1$ стираний в любом столбце. Таким образом, в начале итерации j+1 каждый столбец C содержит 0 или не менее j+2 стираний. □

Вход: $l \times N$ матрица C, соответствующая кодовому слову со стираниями.

Выход: кодовое слово ОКК.

1. *Пусть* $j \leftarrow 0$.

2. *Если есть* $s > j$ *стираний в* i*-ом столбце,* $0 \le i < N$*, матрицы* C*, установить* $X_{j,i} \leftarrow \epsilon$. *Иначе вычислить* $X_{j,i} \leftarrow ((F_l^T)^{-1} C_{-,i})_j$.

3. *Произвести декодирование* $(X_{j,0}, \ldots, X_{j,N-1})$ *с исправлением стираний во внешнем коде* A_j. *В случае неудачи отказаться от декодирования и завершить работу.*

4. *Использовать символы* $X_{t,i}, 0 \le t \le j$*, для исправления стираний в столбцах* i *матрицы* C*, имеющих не более* $j+1$ *стираний, как показано в доказательстве леммы 19.*

5. *Если в матрице* C *остались стирания и* $j < l-1$*, положить* $j \leftarrow j+1$ *и перейти к шагу 2.*

Рис. 6.6. Многошаговое декодирование ОКК с внутренними кодами Рида-Соломона

Теорема 9. *Для ОКК с внутренними кодами Рида-Соломона и внешними* (N, K_i, D_i) *кодами* $\mathbb{C}_i$ *многошаговый алгоритм декодирования способен исправить любую конфигурацию из* $d-1$ *или менее стираний, где* $d = \min_i D_i(i+1)$

Доказательство. Рассмотрим декодирование кодового слова с $d-1$ стираниями. Из леммы 20 вытекает, что для всех i в начале итерации i столбцы матрицы C имеют либо 0, либо не менее $i+1$ стираний. Следовательно, число столбцов со стираниями не превосходит $\left\lfloor \frac{d-1}{i+1} \right\rfloor \le D_i - 1$. Это означает, что декодирование внешнего кода $\mathcal{C}_i$ на шаге 3 будет успешным, т.е. отказ от декодирования не произойдет. После l итераций столбцов со стираниями не останется. □

Многошаговый алгоритм способен исправить и некоторые комбинации из большего числа стираний $d-1$ erasures.

Лемма 21. *Пусть* $E_j \subset \{0, \ldots, N-1\}$ *— некоторые конфигурации стираний, т.е. наборов номеров стертых символов, исправимых внешним кодом* $\mathcal{C}_j, 0 \le j < l$*. Если* $E_0 \supset E_1 \supset \cdots \supset E_{l-1}$*, то многошаговый алгоритм для ОКК с внешними кодами* $\mathcal{C}_j$ *и внутренними кодами Рида-Соломона длины* l *способен исправить любую конфигурацию стираний, в которой* i*-ый столбец матрицы, соответствующей кодовому слову, имеет не более*

$$w(i) = |\{j | i \in E_j\}|, 0 \le i < N,$$

стираний.

Доказательство. Пусть $C \in \mathbb{F}_q^{l\times N}$ — кодовое слово с не более чем $w(i)$ стираниями в столбце i. Ввиду вложенности исправимых конфигураций E_j, имеет место соотношение $w(i) = 1 + \max_{j:i\in E_j} j$. Рассмотрим j-ую итерацию многошагового алгоритма декодирования. На шаге 2 формируется вектор $(X_{j,0},\ldots,X_{j,N-1})$, в котором стирания имеют место в позициях, задаваемых E_j, которые могут быть исправлены декодером внешнего кода. □

Напомним, что полярные коды могут рассматриваться как ОКК, в которых внешние коды также являются полярными. Это позволяет использовать многошаговый алгоритм рекурсивно для декодирования внешних кодов, как показано на рис. 6.7. Этот алгоритм может рассматриваться как реализация многошагового алгоритма или декодера последовательного исключения. Пол узлом F_l здесь понимается устройство, реализующее умножение входного вектора (показанного слева на рис. 6.5) на матрицу F_l.

Сложность этого алгоритма может быть уменьшена, если выполнить некоторые его шаги заранее. В частности, если все входные символы узла являются замороженными, т.е. равными нулю, то все его выходные символы также равны нулю, а значит, могут быть заранее объявлены известными.

Видно, что для каждого узла в схеме кодирования шаги А и Б выполняются не более чем один раз, и число таких узлов равно $\frac{n}{l}\log_l n$. Сложность непосредственной реализации шагов А и Б составляет $O(l^2)$. Таким образом, сложность представленного алгоритма равна $O(nl\log_l n)$.

Описанный алгоритм может рассматриваться как реализация метода последовательного исключения и имеет большое сходство с алгоритмом кодирования обобщенных кодов с локализацией ошибок [116].

6.5.1.2. Эффективная реализация шага А

Рассмотрим случай $l = q = 2^\mu$, и предположим не ограничивая общности, что $\alpha_0 = 0$ и $\alpha_i = \alpha^{q-1-i}, 0 \le i < q-1, \alpha_{q-1} = 0$ в (1.18). Можно заметить, что

$$F_l = \begin{pmatrix} & & & 0 \\ & W & & \vdots \\ & & & 0 \\ 1 & 1 & \ldots & 1 & 1 \end{pmatrix}, F_l^{-1} = \begin{pmatrix} & & & 0 \\ & W^{-1} & & \vdots \\ & & & 0 \\ 1 & 0 & \ldots & 0 & 1 \end{pmatrix},$$

где $W_{ij} = \alpha^{(q-1-i)j}$ и $Q = (W^{-1})^T$ — матрица дискретного преобразования Фурье, т.е. $Q_{ij} = \alpha^{ij}$. Тогда шаг А, т.е. вычисление $(S'_0,\ldots,S'_{\delta-1})^T = \tilde{F}_l(P_0,\ldots,P_{q-1})^T$, где S'_i и P_j — входные и выходные символы узла, δ — число требуемых входных символов и $\tilde{F}_l$ — подматрица $(F_l^{-1})^T$,

Вход: кодовое слово c со стираниями.

Выход: кодовое слово.

1. *Объявить все стертые символы c_i "неизвестными", а нестертые — "известными". Объявить все замороженные символы $u_i, i \in \mathcal{F}$, известными.*

2. *Если некоторый узел F_l имеет $t > 0$ неизвестных выходных символов, объявить его входные символы с номерами $0, \ldots, t-1$ неизвестными, если они не были ранее объявлены известными. Объявить оставшиеся его входные символы "требуемыми", если они не были объявлены ранее известными. Проделать эти действия для каждого слоя поляризующего преобразования.*

3. *Повторять, пока все символы не станут известными:*

   А: Если выходные символы узла известны, вычислить его требуемые входные символы (т.е. умножить вектор выходных символов узла на соответствующую подматрицу F_l^{-1}) и объявить их известными.

   Б: Если узел имеет t известных входных символов, t неизвестных и $l-t$ известных выходных символов, восстановить неизвестные символы путем декодирования кода, задаваемого соответствующими строками F_l, как описано в доказательстве леммы 19 (см. также раздел 6.5.2.2). Объявить восстановленные выходные символы известными.

   В: Если узел на входном слое поляризующего преобразования имеет хотя бы один неизвестный входной символ, объявить отказ от декодирования.

Рис. 6.7. Итеративный алгоритм исправления стираний в полярном коде с ядром Рида-Соломона

соответствующая δ последним строкам ядра, может быть выполнено с помощью спектральных методов вычисления синдрома, описанных в разделе 6.2 со сложностью $O(\delta \log\log l)$ умножения и не более чем $O(\delta l/\log l)$ сложений.

6.5.2. Быстрое систематическое кодирование

Вышеописанный многошаговый/итеративный алгоритм исправления стираний может быть использован для исправления стираний. Действительно, предположим, что все $n-k=|\mathcal{F}|$ проверочных символов стерты, и попытаемся их восстановить с помощью вышеописанного алгоритма. Местоположение проверочных символов должно быть выбрано таким образом, чтобы этот алгоритм был способен их восстановить. Далее предлагается способ такого выбора их местоположения, который позволяет дополнительно снизить вычислительную сложность.

В разделе 6.5.2.1 представлено обобщение метода, приведенного в [31]. В разделе 6.5.2.2 представлена улучшенная реализация шага B в итеративном алгоритме исправления стираний.

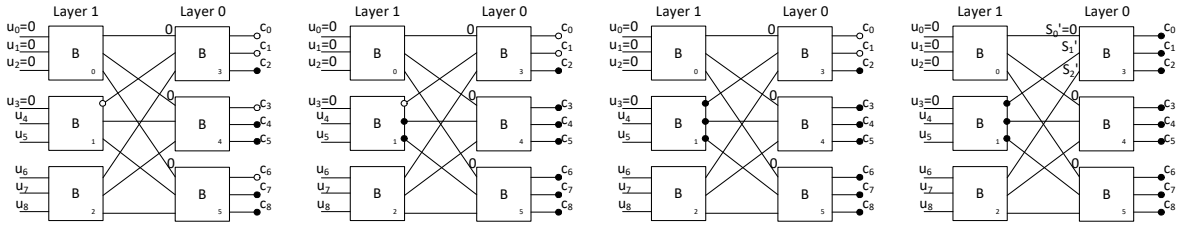
6.5.2.1. Расположение проверочных символов

Следующая теорема показывает, что проверочные символы могут быть расположены в позициях кодового слова $R(j), j \in \mathcal{F}$, где $R(j)$ — функция обращения цифр. Для того, чтобы обосновать это, будем рекурсивно представлять полярные коды как ОКК. На уровне τ рекурсии получив полярный код с множеством номеров замороженных символов $\mathcal{F}^{(p)} \subset \{0,\ldots,l^{m-\tau}-1\}, p \in \{0,\ldots,l-1\}^{\tau}$ будем представлять его как ОКК с внешними полярными кодами с множеством номеров замороженных символов $\mathcal{F}^{(p,t)} = \left\{i | 0 \le i < l^{m-\tau-1}, (tl^{m-\tau-1}+i) \in \mathcal{F}^{(p)}\right\}, 0 \le t < l$. Здесь p обозначает τ-мерный номер кода.

Теорема 10. *Рассмотрим полярный код длины $n = l^m$ с ядром Рида-Соломона и множеством номеров замороженных символов $\mathcal{F}$, причем $\mathcal{F}^{(p,t)} \supset \mathcal{F}^{(p,t+1)}$ для любого $p \in \cup_{i=0}^{m-2}\{0,\ldots,l-1\}^i$, $0 \le t < l-1$. Если стертыми являются только символы кодового слова $c_{R(j)}, j \in \mathcal{F}$, то многошаговый/итеративный алгоритм способен их восстановить.*

Доказательство. Для $m=1$ теорема утверждает, что одноуровневый (l,κ) полярный код с замороженными символами $\mathcal{F}=\{0,1,\ldots,l-\kappa-1\}$, т.е. код Рида-Соломона, способен исправить некоторую конфигурацию из $l-\kappa$ стираний, что, очевидно, верно.

Предположим, что теорема выполняется для некоторого $m \ge 1$, т.е. полярный код длины l^m с множеством номеров замороженных символов $\mathcal{F}$, удовлетворяющим указанным ограничениям, способен исправить конфигурацию стираний $E = \{R(j) | j \in \mathcal{F}\}$. Рассмотрим поляр-



(*а*) (*б*) (*в*) (*г*)

Рис. 6.8. Пример исправления стираний

ный код длины l^{m+1}. Он может быть представлен как ОКК с внутренними кодами Рида-Соломона длины l и внешними полярными кодами $\mathcal{C}_t$ длины l^m с множествами номеров замороженных символов $\mathcal{F}^{(t)}, 0 \le t < l$. Заметим, что каждое из множества $\mathcal{F}^{(t)}$ удовлетворяет условию теоремы.

Всякое кодовое слово C рассматриваемого кода может быть представлено как $l \times l^m$ матрица. Рассмотрим некоторый стертый символ $c_j, j = \sum_{s=0}^{m} j_s l^{m-s}$, где $\sum_{s=0}^{m} j_s l^s \in \mathcal{F}, 0 \le j_s < l$. В матричном представлении кодового слова он находится в столбце $j' = \sum_{s=0}^{m-1} j_s l^{m-1-s}$ и строке j_m. Следовательно, строка t имеет стирания в столбцах из множества $E_t = \left\{R(j) | j \in \mathcal{F}^{(t)}\right\}, 0 \le t < l$. В силу индукционного предположения, конфигурация стираний E_t исправима внешним кодом $\mathcal{C}_t$. Т.к. $\mathcal{F}^{(t)} \supset \mathcal{F}^{(t+1)}, 0 \le t < l-1$, общее число стираний в столбце j' равно $w(j')$. Из леммы 21 вытекает, что многошаговый алгоритм декодирования способен исправить все стирания в C.

Таким образом, многошаговый алгоритм декодирования способен исправить конфигурацию стираний E в коде длины l^{m+1}. □

Эта теорема позволяет задать местоположение проверочных символов при систематическом кодировании полярных кодов с ядром Рида-Соломона.

Пример 19. *Рассмотрим полярный код, соответствующий схеме кодирования, приведенной на рис. 6.5. Здесь* $\mathcal{F} = \{0,1,2,3\}$, *т.е.* $\mathcal{F}^{(0)} = \{0,1,2\}, \mathcal{F}^{(1)} = \{0\}, \mathcal{F}^{(2)} = \varnothing$. *Эти множества удовлетворяют условиям теоремы 10, поэтому проверочные символы могут быть размещены в позициях* $0,3,6,1$.

Эти проверочные символы могут быть восстановлены следующим образом. Заметим, что выход узла 0 *всегда равен нулю. Поэтому,* 0*-ый входные символы узлов* $3,4,5$ *известны. Объявим символы* c_0, c_1, c_3, c_6 *неизвестными, а* c_2, c_4, c_5, c_7, c_8 *— известными. На рис. 6.8, а известные символы показаны черными кругами, а неизвестные — белыми. Таким образом, входной символ* 1 *узла* 3 *должен быть объявлен неизвестным.*

Теперь узлы 4 и 5 имеют один известный входной символ (равный 0) и один неизвестный выходной символ. Следовательно, c_3 и c_6 могут быть восстановлены путем декодирования с исправлением стираний векторов (ϵ, c_4, c_5) и (ϵ, c_7, c_8) в $(3,2,2)$ коде Рида-Соломона. Теперь все выходные символы узлов 4 и 5 известны, что позволяет вычислить все их входные символы (в действительности нужен только символ 1) и объявить их известными, как показано на рис. 6.8, б.

После этого узел 1 имеет один известный входной символ, два известных и один неизвестный выходной символ. Последний (т.е. входной символ 1 узла 3) может быть восстановлен путем декодирования с исправлением стираний в $(3,2,2)$ коде Рида-Соломона, как показано на рис. 6.8, в.

Наконец, c_0 и c_1 могут быть найдены путем декодирования с исправлением стираний в смежном классе $(3,1,3)$ кода Рида-Соломона, который задается символом 1 узла 3. Рассмотрим подробнее эту операцию. Входные символы S'_0, S'_1, S'_2 узла 3 связаны с выходными символами соотношением $(S'_0, S'_1, S'_2) = (c_0, c_1, c_2)Q$, где $Q = F_l^{-1}$. Отсюда получим

$$S'_0 = c_0 Q_{00} + c_1 Q_{10} + c_2 Q_{20} \tag{6.26}$$

$$S'_1 = c_0 Q_{01} + c_1 Q_{11} + c_2 Q_{21} \tag{6.27}$$

$$S'_2 = c_0 Q_{02} + c_1 Q_{12} + c_2 Q_{22}. \tag{6.28}$$

(6.26)–(6.27) *представляют собой уравнения синдромного декодирования кода с проверочной матрицей, равной двум первым строкам матрицы Q^T. Уравнение (6.28) является лишним и не может быть использовано, т.к. S'_2 не может быть вычислен без знания значений u_6, u_7, u_8, которые недоступны декодеру. Зная значения c_2, S'_0, S'_1, можно найти из (6.26)–(6.27) стертые символы c_0, c_1.*

Стандартным способом построения полярных кодов является выбор в качестве $\mathcal{F}$ множества номеров подканалов $W_m^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i)$, задаваемых поляризующим преобразованием, с наибольшими значениями параметров Бхаттачарьи $Z_m^{(i)}$. В настоящее время отсутствуют методы их расчета для случая ядра Рида-Соломона и произвольного канала. Но если $W(y|c)$ является q-ичным стирающим каналом, то подканлы $W_m^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i)$ также являются таковыми, а их параметры Бхаттачарьи, совпадающие с вероятностью стирания, могут быть выражены через вероятность отказа от декодирования соответствующих кодов Рида-Соломона как [125]

$$Z_m^{(i)} = \sum_{t=i''+1}^{l} \binom{l}{t} \left(Z_{m-1}^{(i')}\right)^t \left(1 - Z_{m-1}^{(i')}\right)^{l-t}, \tag{6.29}$$

где $i = i'' + li'$, $0 \le i'' < l$, и $Z_0^{(0)}$ — вероятность стирания в $W(y|c)$. Видно, что $Z_v^{(i''+li')} \ge$

$Z_v^{(i''+1+li')}, i'' < l-1$ для всех $v > 0$. Отсюда и из леммы 4.7 [101] вытекает, что полярные коды с ядром Рида-Соломона, построенные для q-ичного стирающего канала, удовлетворяют требованиям теоремы 10. Действительно, параметры Бхаттачарьи подканалов, получаемых из $\mathcal{W}_v^{(i''+li')}$, не могут быть меньше параметров Бхаттачарьи аналогичных подканалов, получаемых из $\mathcal{W}_v^{(i''+1+li')}, i'' < l-1$.

Более подробная характеристика множества исправимых конфигураций стираний задается следующей теоремой.

Теорема 11. *Рассмотрим некоторую конфигурацию стираний, исправимую итеративным/многошаговым алгоритмом. Пусть $E_{\lambda,j}$ — множество неизвестных выходных символов узла j на некотором слое λ после шага 2 алгоритма. Если E' — другая конфигурация стираний, для которой $|E'_{\lambda,j}| = |E_{\lambda,j}|, 0 \le j < l^{m-1}$, то E' также исправима.*

Доказательство. Множество неизвестных входных символов всех узлов на слое λ после шага 2 является одинаковым для обеих конфигураций стираний. Т.к. эти входные символы восстановимы для E, они восстановимы и для E'. Следовательно, условие на шаге С итеративного алгоритма никогда не выполнится, и он успешно завершит работу за конечное число шагов. □

6.5.2.2. Эффективная реализация шага Б

В данном разделе представлено описание эффективной реализации шага Б, которая становится возможной в тех случаях, когда множество стертых выходных символов, восстанавливаемых отдельными узлами, может быть выбрано заранее, как в случае систематического кодирования с помощью декодирования.

Ограничим $l = q = 2^\mu$. Пусть $(S_0, \dots, S_{l-1})$ и $(P_0, \dots, P_{l-1})$ — соответственно входные и выходные символы некоторого узла, и $E = \{e_1, \dots, e_\delta\}, \delta < l$ — множество номеров неизвестных выходных символов. Предположим, что $\alpha_i = \alpha^{q-1-i}, 0 \le i < q-1, \alpha_{q-1} = 0$. Тогда получим, что

$$S_i + \underbrace{\sum_{j \notin E} \alpha_j^i P_j}_{S'_i} = \sum_{j=1}^{\delta} \alpha_{e_j}^i P_{e_j}, 0 \le i < \delta.$$

Значения S'_i могут быть найдены с помощью процедуры вычисления синдрома, основанной на БПФ. Тогда стертые символы $P_{e_1}, \dots, P_{e_\delta}$ могут быть получены с помощью метода, описанного в разделе 6.3.

Если E представляет собой объединение циклотомических классов $\{a_i, 2a_i, \dots, 2^{m_i-1}a_i\}$, где $2^{m_i}a_i \equiv a_i \bmod 2^\mu - 1$ и $m_i | \mu$, то $\Lambda(x)$ является многочленом с двоичными коэффициен-

тами. В этом случае вычисление (1.26) не требует умножений. Кроме того, (1.5.2) может быть вычислено с помощью циклотомического алгоритма БПФ, как описано в разделе 6.3.3. Заметим также, что вычисление (1.26) эквивалентно умножению двоичной теплицевой матрицы $\tilde{\Lambda}$, содержащей величины Λ_i, на вектор значений $S_i + S'_i$. Следовательно, неизвестные выходные символы $P_{e_1}, \dots, P_{e_\delta}$ узла могут быть вычислены как

$$\begin{pmatrix} P_{e_1} \\ \vdots \\ P_{e_\delta} \end{pmatrix} = \underbrace{\operatorname{diag}\left(\frac{\alpha_{e_i}}{\Lambda'(\alpha^{-e_i})}, 1 \le i \le \delta\right)}_{D} LA\tilde{\Lambda} \begin{pmatrix} S_0 + S_{0'} \\ \vdots \\ S_{\delta-1} + S'_{\delta-1}, \end{pmatrix}, \tag{6.30}$$

где L — блочно-диагональная матрица, состоящая из циркулянтных матриц L_i.

Таким образом, сложность шага B может быть существенно снижена, если неизвестные выходные символы отображаются на объединение циклотомических классов. Это всегда возможно сделать, если $\mu = 2^\tau$.Т.к. $m_i|\mu$, можно построить разложение $\delta = \delta'\mu + \sum_{i=0}^{\tau-1} \delta_i m_i, \delta_i \in \{0,1\}$, и выбрать E как объединение циклотомических классов мощности μ и $m_i : \delta_i = 1$.

Пример 20. *Рассмотрим случай* $l = 16, m = 2, \mathcal{F} = \{0,1,2,3,4,16,17,18\}$, $\alpha_i = \alpha^i, 0 \le i < 15, \alpha_{15} = 0$, *т.е. имеются* 5 *и* 3 *замороженных символов на входе узлов* 0 *и* 1 *на слое* 1. *Целесообразно отобразить стертые символы на выходе узла* 0 *на позиции* $0, 1, 2, 4, 8$, *а для узла 1 — на позиции* $0, 5, 10$. *В случае узла* 1 *получим многочлен локаторов стираний* $\Lambda(x) = (1+x)(1+\alpha^5 x)(1+\alpha^{10}x) = 1+x^3$, $\Lambda'(x) = x^2$, *так что* D *и* $\tilde{\Lambda}$ *в* (6.30) *становятся единичными матрицами и* $L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \alpha^5 & 1+\alpha^5 \\ 0 & 1+\alpha^5 & \alpha^5 \end{pmatrix}$. *Вычисление произведения этой матрицы и вектора требует лишь* 1 *умножения на* α^5, *в то время как непосредственное вычисление* (1.5.2) *требует 9 умножений. В случае узла 0 восстановление 5 выходных символов требует* 5 *умножений.*

Таким образом, предлагаемый подход позволяет производить систематическое кодирование полярных кодов с ядром Рида-Соломона со сложностью $O(n \log n \log^t l), t \le 2$. Это совпадает с асимптотической сложностью несистематического кодирования, если оно реализуется с помощью БПФ. Однако предложенный метод систематического кодирования требует использования неполного БПФ, т.е. вычисления нескольких компонент ДПФ, что намного проще.

Пример 21. *Для* $(256, 248)$ *кода, рассмотренного в примере 20, предложенный алгоритм требует 89 умножений. Это намного меньше чем сложность систематического кодирова-*

ния по определению (1976 умножение) и несистематического кодера, использующего БПФ для умножения на матрицу ядра (512 умножений).

6.6. Ускоренный поиск корней многочленов над конечными полями

Классические и списочные алгоритмы декодирования кодов Рида-Соломона включают этап поиска корней многочленов над конечным полем. В данном разделе представлены быстрые вычислительные процедуры решения этой задачи.

6.6.1. Аффинное разложение

В данном разделе описывается улучшенный алгоритм вычисления значений многочлена в большом числе точек, который может быть использован при переборном поиске корней [64].

Определение 11. *Линеаризованными многочленами над* $\mathbb{F}_{2^m}$ *называются многочлены вида*

$$l(x) = \sum_i l_i x^{2^i}, l_i \in \mathbb{F}_{2^m}$$

Несложно показать, что для линеаризованных многочленов выполняется

$$l(a+b) = l(a) + l(b),\ a, b \in \mathbb{F}_{2^m}. \tag{6.31}$$

Следствием этого свойства является следующая теорема, представленная здесь в немного модифицированном виде:

Теорема 12 ([4])**.** *Пусть* $y \in \mathbb{F}_{2^m}$ *и элементы* $\beta_0, \beta_1, ..., \beta_{m-1}$ *образуют некоторый базис этого поля. Если*

$$y = \sum_{i=0}^{m-1} y_i \beta_i, y_i \in \mathbb{F}_2,$$

то

$$l(y) = \sum_{i=0}^{m-1} y_i l(\beta_i).$$

Обобщением линеаризованных многочленов являются аффинные многочлены, т.е. многочлены вида $a(x) = l(x) + b, b \in \mathbb{F}_{2^m}$, где $l(x)$ — линеаризованный многочлен.

Стандартным приемом снижения сложности вычислительных процедур является повторное использование на каждом шаге результатов ранее выполненных вычислений. В случае линеаризованных и аффинных многочленов для этого может быть использовано свойство (6.31).

Действительно, предположим, что необходимо вычислить набор значений аффинного многочлена $A(x) = L(x) + \beta$ во всех точках $x^{(i)} \in \mathbb{F}_{2^m}$. В силу теоремы 12, эта задача может быть представлена как

$$A(x^{(i)}) = \beta + \sum_{j=0}^{m-1} x_j^{(i)} L(\alpha^j),\ x^{(i)} = \sum_{j=0}^{m-1} x_j^{(i)} \alpha^j, x_j^{(i)} \in \mathbb{F}_2\,. \tag{6.32}$$

Тогда

$$A(x^{(i)}) = A(x^{(i-1)}) + \sum_{j: x_j^{(i)} \neq x_j^{(i-1)}} L(\alpha^j), x^{(0)} = 0. \tag{6.33}$$

Выражение (6.33) задает процедуру вычислений, приводящую к набору значений многочлена $A(x)$ во всех $x^{(i)} \in \mathbb{F}_{2^m}$. Для своего функционирования он требует наличия значений $L(\alpha^j)$, которые должны быть найдены на подготовительном этапе вычислений.

Для минимизации сложности этой процедуры необходимо найти такое упорядочение элементов конечного поля, при котором число сложений на каждом шаге было бы минимально, т.е. необходимо найти такую последовательность элементов конечного поля $x^{(i)}$, что $\sum_{i=1}^{2^m-1} |\{j | x_j^{(i)} \neq x_j^{(i-1)}\}|$ минимально. При этом отметим, что каждое из слагаемых в этом выражении не может быть меньше единицы.

Определение 12. *Кодом Грея называется такая последовательность двоичных векторов длины m, что соседние вектора различаются ровно в одной позиции.*

В силу того, что каждому элементу $\mathbb{F}_{2^m}$ может быть сопоставлен двоичный вектор длины m, элементы поля всегда могут быть упорядочены в соответствии с каким-либо кодом Грея. Поэтому на каждом шаге процедуры (6.33) придется выполнять ровно одно сложение. В настоящее время известны эффективные методы порождения кода Грея [20], что делает описанную процедуру чрезвычайно эффективной.

Описанный подход может быть обобщен на случай произвольных многочленов, если удастся разбить их на набор аффинных.

Теорема 13. *Всякий многочлен $f(x) = \sum_{i=0}^{t} f_i x^i$ может быть разложен на сумму многочленов, кратных аффинным:*

$$f(x) = f_3 x^3 + \sum_{i=0}^{\lceil (t-4)/5 \rceil} x^{5i}(f_{5i} + L_i(x)), \tag{6.34}$$

где $L_i(x) = \sum_{j=0}^{3} f_{5i+2^j} x^{2^j}$ — линеаризованные многочлены, $f_j = 0$ для всех $j > t$

Доказательство. Пусть $k = \lceil (t-4)/5 \rceil$. Выражение (6.34) может быть переписано как

$$\begin{aligned} f(x) = \quad f_k(x) = \quad & f_3 x^3 + \sum_{i=0}^{k} x^{5i}(f_{5i} + L_i(x)) = \\ & F_{k-1}(x) + x^{5k}(f_{5k} + f_{5k+1}x + f_{5k+2}x^2 + f_{5k+4}x^4) + \\ & x^{5(k-1)} f_{5k+3} x^8, \end{aligned}$$

где $f_{-1}(x) = 0$. Для $k = 0\,(t \le 4)$ справедливость теоремы очевидна. Предположим, что она выполняется для некоторого $k - 1 > 0$. Тогда очевидно, что предпоследнее слагаемое (6.35) является многочленом, кратным аффинному, а последнее может быть сгруппировано с соответствующим слагаемым $f_{k-1}(x)$. □

Таким образом, для вычисления значения многочлена во всех точках конечного поля может использоваться следующий алгоритм:

1. Построить таблицы значений линеаризованных многочленов из разложения (6.34): $L_i^{(k)} = L_i(\alpha^k)$, $k = [0; m-1]$, $i \in [0; \lceil (t-4)/5 \rceil]$;
2. Произвести инициализацию $A_i^{(0)} = f_{5i}$;
3. Представив все элементы поля $x_j \in \mathbb{F}_{2^m}$, $j \in [0; 2^m - 1]$, разложенные в стандартном базисе, как элементы кода Грея, вычислить $A_i^{(j)} = A_i^{(j-1)} + L_i^{(\delta(x_j, x_{j-1}))}, j \in [1; 2^m - 1]$, где $\delta(x_j, x_{j-1})$ указывает координату, в которой x_j отличается от x_{j-1};
4. Вычислить $f(x_j) = f_3 x_j^3 + \sum_{i=0}^{\lceil (t-4)/5 \rceil} x_j^{5i} A_i^{(j)}, j \in [0; 2^m - 1]$. Если $f(x_j) = 0$, x_j является корнем многочлена.

Сложность данного алгоритма равна

$$C_{aff} = m \left\lceil \frac{t+1}{5} \right\rceil (4C_{mul} + 3C_{add}) + \left(\left\lceil \frac{t+1}{5} \right\rceil (2C_{add} + C_{mul}) + 2C_{exp} \right) W, \tag{6.35}$$

где W — число обращений к процедуре вычисления значений многочлена, C_{mul} — сложность операции умножения и C_{add} — сложность операции сложения. W может быть меньше 2^m, т.к. вычисления могут быть остановлены досрочно при нахождении всех t корней.

Необходимо отметить, что впервые подход, основанный на сведении произвольного многочлена к набору аффинных, был предложен в работе [181]. Однако описанное там преобразование было возможно только для некоторых многочленов степени не выше 11 и требовало громоздких предварительных вычислений. По имеющейся у автора информации, применение кода Грея для вычисления значений аффинных многочленов в литературе описано не было.

6.6.2. Специальные разложения

В некоторых случаях оказывается возможным построить более эффективные алгоритмы на основе других разложений.

Пример 22. *Для многочленов степени $t \le 8$ может быть использовано разложение*

$$\begin{aligned} F(x) &= \sum_{i=0}^{8} f_i x^i = A_1(x) + x^3(A_2(x) + f_6 x^3), \\ A_1(x) &= f_0 + L_1(x) = f_0 + f_1 x + f_2 x^2 + f_4 x^4 + f_8 x^8, \\ A_2(x) &= f_3 + L_2(x) = f_3 + f_5 x^2 + f_7 x^4. \end{aligned} \tag{6.36}$$

Использование этого разложения позволяет избежать выполнения одной операции возведения в степень. Сложность вычислений с использованием этого разложения равна

$$C_8 = m(6C_{mul} + 4C_{add}) + (4C_{add} + 2C_{mul} + C_{exp})\, W. \tag{6.37}$$

Пример может быть обобщен на случай $t \le 17$:

Пример 23.

$$\begin{aligned} F(x) &= A_1'(x) + x^3(A_2(x) + x^3(f_6 + x^3(A_3(x) + x^3(A_4(x) + f_{15}x^3)))), \\ A_3(x) &= f_9 + f_{10}x + f_{11}x^2 + f_{13}x^4 + f_{17}x^8, \\ A_4(x) &= f_{12} + f_{14}x^2, \end{aligned} \tag{6.38}$$

Сложность вычислений с использованием этого разложения равна

$$C_{17} = m(12C_{mul} + 8C_{add}) + (9C_{add} + 5C_{mul} + C_{exp})\, W. \tag{6.39}$$

6.6.3. Обобщенное разложение

Одним из возможных методов построения разложений, отличных от (6.34), является замена показателя степени 5 (эта величина будет называться далее показателем разложения) другим числом. Рассмотрим ограничения, которым должен удовлетворять показатель разложения.

Определение 13. *Число r называется примитивным корнем по модулю простого числа p, если*

$$(\forall j : 0 < j < p)\, \exists i : 0 \le i < p-1 : r^i \equiv j \pmod{p}$$

Для того, чтобы построить разложение по многочленам, кратным аффинным, необходимо так сгруппировать слагаемые, чтобы выполнялось соотношение:

$$\forall i \notin A\, \exists j, k : i = pj + 2^k, \tag{6.40}$$

где A — множество номеров коэффициентов, группируемых в остаточный член произвольного вида (например, в разложении (6.34) $A = \{3\}$). Это приведет к разложению $F(x) =$

$F_0(x) + \sum_i x^{pi} A_i(x)$, где $F_0(x)$ — некоторый остаточный член, соответствующий множеству A, p — простое число, $A_i(x)$ — аффинные многочлены. Очевидно, что условие (6.40) эквивалентно тому, что число 2 является примитивным корнем по модулю p. Однако оказывается, что для простых чисел, больших 5, не удается построить разложения с небольшим остаточным членом (множеством A малой мощности). Эта трудность может быть преодолена, если учесть, что все ненулевые элементы конечного поля $\mathbb{F}_{2^m}$ образуют циклическую группу по умножению порядка $n = 2^m - 1$.

Предположим, что $n = \prod_i p_i$, где p_i — простые числа, одно из которых p_0 имеет примитивный корень 2. Тогда (6.40) может быть заменено на

$$\forall i \, \exists j, k : \, i \equiv p_0 j + 2^k \bmod n \tag{6.41}$$

В силу того, что 2 является примитивным корнем по модулю p_0, последнее соотношение выполняется для всех $i < n$, т.е. возможно построение разложения с нулевым остаточным членом.

6.6.4. Гибридный алгоритм поиска корней многочленов

Ввиду того, что аналитический алгоритм поиска корней многочленов степени не выше 4 выполним за детерминированное и достаточно малое время, представляется целесообразным использовать его при поиске корней произвольных многочленов. Это может быть выполнено путем деления исходного многочлена степени t на многочлен $\prod_{i=1}^{t-4}(x - x_i)$, где x_i — корни многочлена, которые должны быть найдены каким-либо иным методом. Исследования показали, что наиболее быстрым алгоритмом является метод аффинного разложения (см. п. 6.6.1). Таким образом, можно предложить следующий алгоритм поиска корней многочленов произвольной степени:

1. Если степень многочлена превышает 4, то построить аффинное разложение в соответствии с п. 6.6.1 и перебирать элементы поля, упорядоченные в соответствии с кодом Грея, вычисляя значение многочлена. При обнаружении корня x_i разделить исходный многочлен на $(x - x_i)$. Как только степень многочлена понизится до 4, перейти к аналитическому методу поиска корней.

2. Корни многочленов степени 3 и 4 могут быть найдены аналитически [184]. С другой стороны, они могут быть найдены и вышеописанным способом.

3. Значения корней многочлена степени 2 после линейной подстановки могут быть найдены с помощью метода Берлекэмпа-Рамсея-Соломона [4].

Для оценки реальной эффективности предложенных методов поиска корней многочленов локаторов ошибок они были реализованы программно на языке C++. В таблице 6.6 представлены результаты замеров скорости вычислений, выполненных на ПК с процессором AMD Athlon XP 1700+.

Таблица 6.6. Среднее время (мкс) поиска корней многочленов различными методами

Степень	Алгоритм Ченя	Общее аффинное разложение (6.34)	Специальное разложение (6.36)	Гибридный метод, основанный на (6.36)	Специальное разложение (6.38)	Гибридный метод, основанный на (6.38)
5	14,6	13,4	10,0	2,3	—	—
6	17,2	14,8	10,1	3,6	—	—
7	19,6	14,9	10,8	4,9	—	—
8	22,2	15,1	10,8	5,8	11,4	6,7
9	24,9	15,4	—	—	15,5	9,5
10	27,3	16,9	—	—	16,4	11,0
11	29,8	18,0	—	—	16,4	11,6
12	32,3	18,1	—	—	16,4	12,2
13	34,8	18,2	—	—	16,9	13,5
14	37,3	18,2	—	—	17,4	14,2
15	39,8	20,1	—	—	17,4	14,8
16	42,3	21,3	—	—	17,9	15,8
17	44,9	21,3	—	—	17,9	16,2

Видно, что применение аффинного разложения позволяет достичь выигрыша по быстродействию до 2 раз, а применение специальных разложений в сочетании с аналитическим методом поиска корней может дать шестикратный выигрыш по производительности.

6.7. Выводы

В данной главе был представлен усовершенствованный циклотомический алгоритм быстрого преобразования Фурье, который обеспечивает асимптотически меньшую сложность по сравнению с вариантом, приведенном в [23, 162]. Был также представлен метод построения быстрых процедур умножения фиксированной двоичной матрицы на произвольный вектор,

позволяющий существенно упростить один из этапов циклотомического алгоритма БПФ.

Кроме того, был представлен новый метод быстрого систематического кодирования кодов Рида-Соломона, основанный на предложенном алгоритме БПФ. Было показано, что предложенный подход обеспечивает существенно меньшую сложность по сравнению с классическим методом, а его программная реализация обеспечивает существенно большую производительность по сравнению с широко используемой в системах хранения данных библиотекой Jerasure. Кроме того, было представлено обобщение предложенного метода систематического кодирования на случай полярных кодов с ядром Рида-Соломона. Был также представлен упрощенный метод поиска корней многочленов над полями характеристики 2, который может быть использован для снижения сложности классических и списочных алгоритмов декодирования кодов Рида-Соломона.

Заключение

В диссертационной работе решена научная проблема построения кодов, допускающих простое декодирование с хорошей корректирующей способностью, имеющая важное теоретическое и прикладное значение. Получены следующие основные результаты.

1. Разработана математическая модель линейных блоковых кодов, в основе которой лежит их представление в виде системы линейных ограничений динамического замораживания на входные символы поляризующего преобразования. Оно эквивалентно представлению кода как множества векторов значений многочленов (или векторных многочленов) от нескольких переменных в различных точках. Это представление позволяет использовать для декодирования метод последовательного исключения и его обобщения. Хотя в общем случае вероятность ошибки декодирования произвольного кода методом последовательного исключения превосходит вероятность ошибки при использовании других известных алгоритмов декодирования, при использовании списочного или последовательного алгоритма с достаточно большим размером списка предложенный подход обеспечивает декодирование почти по максимуму правдоподобия. Размер списка, требуемый для этого, зависит от конфигурации множества номеров замороженных символов и, в частности, от максимальной степени многочленов, задающих кодовые слова рассматриваемого кода. В работе охарактеризованы такие множества для расширенных примитивных кодов БЧХ и Рида-Соломона в узком смысле, а также расширенного кода Голея. Показано, что применение предложенного подхода позволяет выполнить декодирование коротких кодов из указанных семейств со сложностью меньшей или сопоставимой с другими известными алгоритмами декодирования.

2. На основе предложенной математической модели разработан метод построения полярных подкодов, который позволяет получить коды произвольной длины, допускающие декодирование с помощью списочного или последовательного алгоритмов последовательного исключения. Было показано, что предложенные коды обеспечивают лучшую корректирующую способность по сравнению с известными турбо- и LDPC кодами и кодами с циклическим замыканием, а при использовании последовательного алгоритма декодирования одновременно обеспечивается и меньшая сложность декодирования. Это позволяет решить проблему снижения энергопотребления мобильных устройств связи.

3. Для решения проблемы высокой задержки декодирования полярных (под)кодов была предложена конструкция звездных полярных подкодов и соответствующее обобщение

списочного алгоритма Тала-Варди. Было показано, что данный подход позволяет сократить число шагов (списочного) алгоритма последовательного исключения в l раз, где l — параметр конструкции.

4. Предложен последовательный алгоритм декодирования полярных подкодов, основанный на учете типичного поведения веса правильного пути в списочном алгоритме последовательного исключения. Было показано, что в худшем случае его сложность не превосходит сложность списочного алгоритма с аналогичной корректирующей способностью, но в среднем сложность с увеличением отношения сигнал/шум приближается к сложности алгоритма последовательного исключения. Использование предложенной математической модели позволяет использовать этот алгоритм для декодирования и иных кодов, представленных в виде системы ограничений динамического замораживания. В частности, было показано, что он позволяет упростить декодирование коротких кодов БЧХ, а для кодов Рида-Соломона обеспечивает повышение корректирующей способности по сравнению с другими известными алгоритмами декодирования. Был предложен метод дальнейшего повышения корректирующей способности при декодировании кодов Рида-Соломона, основанный на использовании различных перестановок зашумленного вектора. Показано, что применение предложенного подхода позволяет выполнить декодирование коротких кодов Рида-Соломона с корректирующей способностью лучшей, чем у известного адаптивного алгоритма распространения доверия.

5. Для решения проблемы эффективного декодирования длинных кодов Рида-Соломона, был разработан быстрый вычислительный алгоритм, реализующий интерполяционный шаг в алгоритмах Гурусвами-Судана и Ву. В его основе лежит двоичный метод возведения в степень и новый рандомизированный алгоритм построения базиса Грёбнера произведения нульмерных идеалов. Применение предложенного подхода обеспечивает значительное асимптотическое и практическое снижение сложности списочного декодирования кодов Рида-Соломона по сравнению с реализацией, исользующей итеративный интерполяционный алгоритм. При наличии ограничений на сложность декодирования, это позволяет увеличить значение параметра указанных алгоритмов, определяющего радиус декодирования, повысив таким образом помехозащищенность существующих и перспективных систем передачи и хранения информации, использующих коды Рида-Соломона.

6. Получена уточненная оценка на требуемую кратность корней и размер списка в алго-

ритме By списочного декодирования кодов Рида-Соломона. Ее использование обеспечивает дополнительное снижение сложности декодирования по сравнению с оригинальным алгоритмом By.

7. Предложен быстрый алгоритм вычисления дискретного преобразования Фурье над полями характеристики 2. В основе предложенного подхода лежит группировка элементов конечного поля по классам сопряженности, их разложение по базисам подполей и использование известных быстрых алгоритмов вычисления циклической свертки и деления с остатком.

8. Предложен быстрый алгоритм систематического кодирования кодов Рида-Соломона, основанный на предложенном циклотомическом алгоритме быстрого преобразования Фурье. Показано, что программная реализация разработанного алгоритма обеспечивает существенно лучшую производительность по сравнению с библиотекой Jerasure, широко используемой в современных системах хранения данных. Полученный результат позволяет увеличить длину кодов Рида-Соломона, используемых для защиты данных от потерь, вызванных отказами дисков, и за счет этого повысить полезную емкость систем хранения данных.

9. Предложен быстрый алгоритм систематического кодирования полярных кодов с ядром Рида-Соломона, основанный на их рекурсивном представлении как обобщенных каскадных и предложенном быстром методе систематического кодирования кодов Рида-Соломона.

10. Предложен упрощенный метод поиска корней многочленов над полями характеристики 2, основанный на представлении многочленов в виде суммы полиномов, кратных аффинным, и упорядочении элементов конечного поля в соответствии с кодом Грея. Предложенный метод может быть использован для повышения производительности классических и списочных декодеров кодов БЧХ и Рида-Соломона.

Список литературы

1. Афанасьев, В. Б. Алгоритмы БПФ для полей $GF(2^m)$ / В. Б. Афанасьев, И. И. Грушко // Помехоустойчивое кодирование и надежность ЭВМ. — М.: Наука, 1987. — С. 33–55.
2. Ахо, А. Построение и анализ вычислительных алгоритмов / А. Ахо, Дж. Хопкрофт, Дж. Ульман. — М.: Мир, 1979.
3. Барг, С. Некоторые новые NP-полные задачи кодирования / С. Барг // Проблемы передачи информации. — 1994. — Т. 30, № 3. — С. 23–28.
4. Берлекэмп, Э. Р. Алгебраическая теория кодирования / Э. Р. Берлекэмп. — М.: Мир, 1971.
5. Блейхут, Р. Теория и практика кодов, контролирующих ошибки / Р. Блейхут. — М.: Мир, 1986.
6. Блейхут, Р. Быстрые алгоритмы цифровой обработки сигналов / Р. Блейхут. — М.: Мир, 1989.
7. Блох, Э. Кодирование обобщенных каскадных кодов / Э.Л. Блох, В.В. Зяблов // Проблемы передачи информации. — 1974. — Т. 10, № 3. — С. 45–50.
8. Блох, Э. Л. Линейные каскадные коды / Э. Л. Блох, В. В. Зяблов. — М.: Наука, 1982. — С. 232.
9. Габидулин, Э. М. Кодирование в радиоэлектронике / Э. М. Габидулин, В. Б. Афанасьев. — М.: Радио и связь, 1986.
10. Галлагер, Р. Теория информации и надежная связь / Р. Галлагер. — М.: Советское радио, 1974.
11. Гриссер, Х. Апостериорно-вероятностное декодирование несистематических блоковых кодов / Х. Гриссер, В.Р. Сидоренко // Проблемы передачи информации. — 2002. — Т. 38, № 3. — С. 20–33.
12. Захарова, Т. Г. Вычисление преобразования Фурье в полях характеристики 2 / Т. Г. Захарова // Проблемы передачи информации. — 1992. — Т. 28, № 2. — С. 62–76.
13. Захарова, Т. Г. Применение преобразования Фурье при декодировании кодов Рида-Соломона / Т. Г. Захарова // Радиотехника. — 1996. — № 12. — С. 55–57.
14. Зигангиров, К. Некоторые последовательные процедуры декодирования / К.Ш. Зигангиров // Проблемы передачи информации. — 1966. — Т. 2, № 4. — С. 13–25.
15. Зиновьев, В. Обобщенные каскадные коды / В.А. Зиновьев // Проблемы передачи информации. — 1976. — Т. 12, № 1. — С. 5–15.
16. Зубков, А. М. Статистические характеристики весовых спектров случайных линейных

кодов над GF(p) / А. М. Зубков, В. И. Круглов // математические вопросы криптографии. — 2014. — Т. 7. — С. 27–38.

17. Колесник, В. Циклические коды Рида-Маллера и их декодирование / В.Д. Колесник, Е.Т. Мирончиков // Проблемы передачи информации. — 1968. — Т. 4, № 4. — С. 20–25.
18. Мак-Вильямс, Ф. Д. Теория кодов, исправляющих ошибки / Ф. Дж. Мак-Вильямс, Н. Дж. А. Слоэн. — М.: Связь, 1979.
19. Милославская, В. Методы построения и декодирования полярных кодов: дисс. канд. техн. наук / Санкт-Петербургский государственный политехнический университет. — 2014.
20. Рейнгольд, Э. Комбинаторные алгоритмы: теория и практика / Э. Рейнгольд, Ю. Нивергельт, Н. Део. — М.: Мир, 1980.
21. Трифонов, П. Интерполяция в списочном декодировании кодов Рида-Соломона / П.В. Трифонов // Проблемы передачи информации. — 2007. — Т. 43, № 3. — С. 28–38.
22. Трифонов, П. Декодирование кодов Рида-Соломона методом последовательного исключения / П.В. Трифонов // Проблемы передачи информации. — 2014. — Т. 50, № 4. — С. 3–14.
23. Трифонов, П. В. Метод быстрого вычисления преобразования Фурье над конечным полем / П. В. Трифонов, С. В. Федоренко // Проблемы передачи информации. — 2003. — Т. 39, № 3. — С. 3–10.
24. Abbe, E. Polar codes for the m-user multiple access channel / Emmanuel Abbe, E. Telatar // IEEE Transactions on Information Theory. — 2012. — August. — Vol. 58, no. 8.
25. Afanasyev, V. On complexity of FFT over finite field / V. Afanasyev // Proceedings of Sixth Joint Swedish-Russian International Workshop on Information Theory, Molle, Sweden. — 1993. — August. — P. 315–319.
26. Alekhnovich, M. Linear Diophantine equations over polynomials and soft decoding of Reed-Solomon codes / Michael Alekhnovich // IEEE Transactions on Information Theory. — 2005. — July. — Vol. 51, no. 7. — P. 2257–2265.
27. Algebraic properties of polar codes from a new polynomial formalism / Magali Bardet, Vlad Dragoi, Ayoub Otmani, Jean-Pierre Tillich // Proceedings of IEEE International Symposium on Information Theory. — 2016.
28. Ali, M. Minimal list decoding of Reed-Solomon codes using a parameterization of Gröbner bases / Mortuza Ali, Margreta Kuijper // Proceedings of IEEE International Symposium on Information Theory. — 2011.
29. Alltop, W. A method for extending binary linear codes / W. Alltop // IEEE Transactions

on Information Theory. — 1984. — November. — Vol. 30, no. 6.

30. Arikan, E. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels / E. Arikan // IEEE Transactions on Information Theory. — 2009. — July. — Vol. 55, no. 7. — P. 3051–3073.
31. Arikan, E. Systematic polar coding / E. Arikan // IEEE Communications Letters. — 2011. — August. — Vol. 15, no. 8. — P. 860–862.
32. Arikan, E. On the rate of channel polarization / E. Arikan, I.E. Telatar // Information Theory, 2009. ISIT 2009. IEEE International Symposium on. — 2009. — June. — P. 1493–1495.
33. Askey, R. A. Generalized hypergeometric function / R. A. Askey, Adri B. Olde Daalhuis // NIST Handbook of Mathematical Functions. — Cambridge University Press, 2010.
34. Research to develop the algebraic theory of codes: Rep. / Sylvania electronic systems, applied research laboratory; Executor: E. F. Assmus, H. F. Mattson, R. J. Turyn: 1967.
35. Balatsoukas-Stimming, A. LLR-based successive cancellation list decoding of polar codes / Alexios Balatsoukas-Stimming, Mani Bastani Parizi, Andreas Burg // IEEE Transactions On Signal Processing. — 2015. — October. — Vol. 63, no. 19. — P. 5165–5179.
36. Barg, A. Error exponents of expander codes under linear-complexity decoding / A. Barg, Gilles Zemor // SIAM Journal on Discrete Mathematics. — 2004. — Vol. 17, no. 3.
37. Baumgartner, B. On iterative multistage decoding for multilevel codes for frequency selective channels / B. Baumgartner, H. Griefier, M. Bossert // Proceedings of International Symposium on Power-line Communications and its Applications. — 2000.
38. Beast decoding for block codes / Irina E. Bocharova, Rolf Johannesson, Boris D. Kudryashov, Maja Loncar // European Transactions Telecommunnications. — 2004. — July. — Vol. 15, no. 4. — P. 297–305.
39. Becker, T. Gröbner Bases. A Computational Approach to Commutative Algebra / T. Becker, V. Weispfenning. — New York: Springer, 1993.
40. Beery, Y. Optimal soft decision block decoders based on fast Hadamard transform / Y. Beery, J. Snyders // IEEE Transactions on Information Theory. — 1986. — May. — Vol. 32, no. 3.
41. Bellini, S. On the structure of cyclotomic Fourier transforms and their applications to Reed-Solomon codes / S. Bellini, M. Ferrari, A. Tomasoni // IEEE Transactions on Communications. — 2011. — August. — Vol. 59, no. 8. — P. 2110 –2118.
42. Berlekamp, E. R. On the inherent intractability of certain coding problems / E. R. Berlekamp, R. J. McEliece, H. C. A. van Tilborg // IEEE Transactions on Information Theory. — 1978. — Vol. 24, no. 5. — P. 384–386.

43. Bhowmick, A. The list decoding radius of reed-muller codes oversmall fields / Abhishek Bhowmick, Shachar Lovett // Proceedings of the 47th annual ACM symposium on Theory of computing. — 2015. — P. 277–285.
44. Blahut, R. E. Transform techniques for error control codes / R. E. Blahut // IBM Journal of Research and Development. — 1979. — May. — Vol. 23, no. 3. — P. 299–315.
45. Canteaut, A. A new algorithm for finding minimum-weight words in a linear code: Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511 / A. Canteaut, F. Chabaud // IEEE Transactions on Information Theory. — 1998. — January. — Vol. 44, no. 1. — P. 367–378.
46. Chung, S.-Y. Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation / S.-Y. Chung, T. J. Richardson, R. L. Urbanke // IEEE Transactions on Information Theory. — 2001. — February. — Vol. 47, no. 2.
47. Coffey, J. T. The capacity of coded systems / John T. Coffey, Aaron B. Kiely // IEEE Transactions On Information Theory. — 1997. — January. — Vol. 43, no. 1.
48. Costa, E. On computing the syndrome polynomial in Reed-Solomon decoder / E. Costa, S. V. Fedorenko, P. V. Trifonov // European Transactions on Telecommunications. — 2004. — May/June. — Vol. 15, no. 4. — P. 337–342.
49. Cox, D. Ideals, varieties and algorithms / D. Cox, G. Little, D. O'Shea. — Springer-Verlag, 1992.
50. Declercq, D. Decoding algorithms for nonbinary LDPC codes over $GF(q)$ / David Declercq, Marc P.C. Fossorier // IEEE Transactions on Communications. — 2007. — April. — Vol. 55, no. 4. — P. 633–643.
51. Delsarte, P. On generalized Reed-Muller codes and their relatives / P. Delsarte, J.M. Goethals, F.J. MacWilliams // Information and control. — 1970. — Vol. 16. — P. 403–442.
52. Dixon, J. D. The degree of the splitting field of a random polynomial over a finite field / John D. Dixon, Daniel Panario // The Electronic Journal of Combinatorics. — 2004. — Vol. 11, no. 1.
53. DMC R&D Center Samsung Electronics Co., L. 5g vision. — 2015. — white paper. http://www.samsung.com/global/business/networks/insights/white-paper.
54. Dumer, I. On complexity of decoding reed-muller codes within their code distance / Ilya Dumer, Grigory Kabatiansky, Cedric Tavernier // Proceedings of International Workshop on Algebraic and Combinatorial Coding Theory. — 2008. — P. 82–85.
55. Dumer, I. Soft-decision list decoding of Reed-Muller codes with linear complexity /

Ilya Dumer, Grigory Kabatiansky, Cedric Tavernier // Proceedings of IEEE International Symposium on Information Theory. — 2011. — P. 2303–2307.

56. Efficient interpolation and factorization in algebraic soft-decision decoding of Reed-Solomon codes / R. Koetter, J. Ma, A. Vardy, A. Ahmed // Proceedings of IEEE International Symposium on Information Theory. — 2003. — P. 365.
57. Ekroot, L. A^* Decoding of Block Codes / L. Ekroot, S. Dolinar // IEEE Transactions on Communications. — 1996. — September. — Vol. 44, no. 9.
58. El-Khamy, M. Iterative algebraic soft-decision list decoding of reed-solomon codes / Mostafa El-Khamy, R.J. McEliece // IEEE Journal on Selected Areas In Communications. — 2006. — March. — Vol. 24, no. 3.
59. List decoding for noisy channels: Rep.: 335 / Research Laboratory of Electronics, MIT; Executor: P. Elias: 1957.
60. Enhanced belief propagation decoding of polar codes through concatenation / Jing Guo, Minghai Qin, Albert Guilleni Fabregas, Paul.H. Siegel // Proceedings of IEEE International Symposium on Information Theory. — 2014.
61. Fast encoding of polar codes with Reed-Solomon kernel / Peter Trifonov, Vera Miloslavskaya, Chen Chen, Yuangang Wang // IEEE Transactions on Communications. — 2016. — July. — Vol. 64, no. 7. — P. 2746–2753.
62. Fast list decoders for polar codes / Gabi Sarkis, Pascal Giard, A. Vardy et al. // IEEE Journal On Selected Areas In Communications. — 2016. — February. — Vol. 34, no. 2. — P. 318–328.
63. Fedorenko, S. A simple algorithm for decoding Reed-Solomon codes and its relation to the Welch-Berlekamp algorithm / S. Fedorenko // IEEE Transactions on Information Theory. — 2005. — March. — Vol. 51, no. 3. — P. 1196–1198.
64. Fedorenko, S. V. Finding roots of polynomials over finite fields / S. V. Fedorenko, P. V. Trifonov // IEEE Transactions on Communications. — 2002. — Vol. 50, no. 11. — P. 1709–1711.
65. Final report of 3gpp tsg ran wg1 #84bis: Rep.: R1-163961 / 3GPP: 2016.
66. Flexible and low-complexity encoding and decoding of systematic polar codes / Gabi Sarkis, Ido Tal, Pascal Giard et al. // IEEE Transactions on Communications. — 2016. — July. — Vol. 64, no. 7.
67. Fontaine, C. How reed-solomon codes can improve steganographic schemes / Caroline Fontaine, Fabien Galand // EURASIP Journal on Information Security. — 2009. — December. — Vol. 2009.
68. Concatenated codes: Rep. / Massachusetts Institute of Technology; Executor: G.D. Forney:

1965. — December.

69. Forum, G. 5g vision, requirements,and enabling technologies. — 2016.
70. Gallager, R. Low-density Parity-Check codes: Ph. D. thesis / MIT. — 1963.
71. Gandikota, V. NP-hardness of Reed-Solomon decoding and the Prouhet-Tarry-Escott problem / Venkata Gandikota, Badih Ghazi, Elena Grigorescu // Proceedings of IEEE 57th Annual Symposium on Foundations of Computer Science. — 2016.
72. Gao, S. A new algorithm for decoding Reed-Solomon codes / Shuhong Gao // Communications, Information and Network Security / Ed. by V. Bhargava, H. V. Poor, V. Tarokh, S. Yoon. — Kluwer, 2003. — P. 55–68.
73. Gohberg, I. Matrix polynomials / I. Gohberg, P. Lancaster, L. Rodman. — SIAM, 2009.
74. Goldsmith, A. J. Wireless Communications / Andrea J. Goldsmith. — Cambridge University Press, 2005.
75. Gopalan, P. List-decoding reed-muller codes over small fields / Parikshit Gopalan, Adam R. Klivans, D. Zuckerman // Proceedings of the fortieth annual ACM symposium on Theory of computing. — 2008. — P. 265–274.
76. Grassl, M. Computing extensions of linear codes using a greedy algorithm / Markus Grassl // Proceedings of IEEE International Symposium on Information Theory. — 2012.
77. Guruswami, V. List Decoding of Error-Correcting Codes: Ph. D. thesis / Massachusets Institute of Technology. — 2001.
78. Guruswami, V. Achieving list decoding capacity using folded Reed-Solomon codes / V. Guruswami, A. Rudra // Proceedings of 44th Allerton Conference on Communication, Control, and Computing. — 2006.
79. Guruswami, V. Better binary list decodable codes via multilevel concatenation / V. Guruswami, Atri Rudra // IEEE Transactions On Information Theory. — 2009. — January. — Vol. 55, no. 1.
80. Guruswami, V. Improved decoding of Reed-Solomon and algebraic-geometric codes / V. Guruswami, M. Sudan // IEEE Transactions on Information Theory. — 1999. — September. — Vol. 45, no. 6. — P. 1757–1767.
81. Guruswami, V. Maximum-likelihood decoding of reed-solomon codes is np-hard / V. Guruswami, A. Vardy // IEEE Transactions On Information Theory. — 2005. — July. — Vol. 51, no. 7.
82. Ha, J. Rate-compatible puncturing of low-density parity-check codes / Jeongseok Ha, Jaehong Kim, Steven W. McLaughlin // IEEE Transactions on Information Theory. — 2004. — November. — Vol. 50, no. 11.

83. Hardware architectures for successive cancellation decoding of polar codes / Camille Leroux, Ido Tal, A. Vardy, W.J. Gross // Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing. — 2011. — May. — P. 1665–1668.
84. Hart, P. E. A formal basis for the heuristic determination of minimum cost paths / Peter E. Hart, Nils J. Nilsson, Bertram Raphael // IEEE Transactions Of Systems Science And Cybernetics. — 1968. — July. — Vol. 4, no. 2.
85. Hassani, S. H. Finite-length scaling for polar codes / S. Hamed Hassani, Kasra Alishahi, R.L. Urbanke // IEEE Transactions On Information Theory. — 2014. — October. — Vol. 60, no. 10.
86. Hassani, S. H. On the scaling of polar codes: I. the behavior of polarized channels / S. Hamed Hassani, R.L. Urbanke // Proceedings of IEEE International Symposium on Information Theory. — 2010.
87. Hussami, N. Performance of polar codes for channel and source coding / Nadine Hussami, Satish Babu Korada, R.L. Urbanke // Proceedings of IEEE International Symposium on Information Theory. — 2009. — P. 1488–1492.
88. Imai, H. A new multilevel coding method using error correcting codes / H. Imai, S. Hirakawa // IEEE Transactions on Information Theory. — 1977. — May. — Vol. 23, no. 3. — P. 371–377.
89. Introduction to Algorithms / Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. — 2 edition. — The MIT Press, 2001.
90. Isaka, M. On the iterative decoding of multilevel codes / Motohiko Isaka, H. Imai // IEEE Journal on Selected Areas in Communications. — 2001. — May. — Vol. 19, no. 5.
91. Ivanov, K. Hybrid decoding of interlinked generalized concatenated codes / K. Ivanov, P. Trifonov // Proceedings of 9th International Symposium on Turbo Codes and Iterative Information Processing. — Brest, France: IEEE, 2016.
92. Johannesson, R. Fundamentals of Convolutional Coding / Rolf Johannesson, K.Sh. Zigangirov. — IEEE Press, 1998.
93. Kasami, T. New generalizations of the Reed-Muller codes part I: Primitive codes / Tadao Kasami, Shu Lin, Wesley Peterson // IEEE Transactions on Information Theory. — 1968. — March. — Vol. 14, no. 2. — P. 189–199.
94. Kasami, T. Polynomial codes / Tadao Kasami, Shu Lin, Wesley Peterson // IEEE Transactions on Information Theory. — 1968. — November. — Vol. 14, no. 6.
95. Kern, D. A new code construction for polar codes using min-sum density / Daniel Kern, Sebastian Vorkoper, V. Kuhn // Proceedings of International Symposium on Turbo Codes

and Iterative Information Processing. — 2014. — P. 228–232.

96. Knuth, D. E. The Art of Computer Programming / D. E. Knuth. — Addison-Wesley, 1973. — Vol. 2.
97. Koetter, R. Fast generalized minimum-distance decoding of algebraic-geometry and Reed-Solomon codes / Ralf Koetter // IEEE Transactions on Information Theory. — 1996. — May. — Vol. 42, no. 3.
98. Koetter, R. Algebraic soft-decision decoding of Reed-Solomon codes / R. Koetter, A. Vardy // IEEE Transactions on Information Theory. — 2003. — November. — Vol. 49, no. 11. — P. 2809–2825.
99. Koetter, R. A complexity reducing transformation in algebraic list decoding of Reed-Solomon codes / R. Koetter, A. Vardy // Proceedings of ITW2003. — 2003. — March. — P. 10–13.
100. Kolesnik, V. Cyclic Reed-Muller codes and their decoding / V.D. Kolesnik, E.T. Mironchikov // Problems of Information Transmission. — 1968. — Vol. 4, no. 4. — P. 15–19.
101. Korada, S. B. Polar codes for channel and source coding: Ph. D. thesis / Ecole Polytechnique Federale de Lausanne. — 2009.
102. Korada, S. B. Polar codes: Characterization of exponent, bounds, and constructions / S. B. Korada, E. Sasoglu, R.L. Urbanke // IEEE Transactions on Information Theory. — 2010. — December. — Vol. 56, no. 12. — P. 6253–6264.
103. Kudekar, S. Spatially coupled ensembles universally achieve capacity under belief propagation / Shrinivas Kudekar, T.J. Richardson, R.L. Urbanke // Proceedings of IEEE International Symposium on Information Theory. — 2012.
104. Lee, K. An interpolation algorithm using Gröbner bases for soft-decision decoding of Reed-Solomon codes / Kwankyu Lee, Michael E. O'Sullivan // Proceedings of IEEE International Symposium on Information Theory. — 2006. — P. 2032 – 2036.
105. Lee, K. List decoding of Reed-Solomon codes from a Gröbner basis perspective / Kwankyu Lee, Michael E. O'Sullivan // Journal of Symbolic Computation. — 2008. — September. — Vol. 43, no. 9.
106. Lehmann, F. Analysis of the iterative decoding of LDPC and product codes using the Gaussian approximation / F. Lehmann, G. M. Maggio // IEEE Transactions on Information Theory. — 2003. — November. — Vol. 49, no. 11. — P. 2993–3000.
107. Leont'ev, V. Roots of random polynomials over a finite field / V.K. Leont'ev // Mathematical Notes. — 2006. — Vol. 80, no. 2. — P. 300–304. — Translated from Matematicheskie Zametki, vol. 80, no. 2, 2006, pp. 313–316.

108. Li, B. An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check / Bin Li, Hui Shen, D.N.C. Tse // IEEE Communications Letters. — 2012. — December. — Vol. 16, no. 12. — P. 2044–2047.

109. Liberty, E. The mailman algorithm: a note on matrix vector multiplication / Edo Liberty, Steven W. Zucker // Information Processing Letters. — 2009. — Vol. 109, no. 3.

110. Liesenfeld, B. On the equivalence of some generalized concatenated codes and extended cyclic codes / B. Liesenfeld, B.G. Dorsch // Proceedings of IEEE International Symposium on Information Theory. — 1993. — P. 399.

111. Lin, S.-J. Novel polynomial basis and its application to reed-solomon erasure codes / Sian-Jheng Lin, Wei-Ho Chung, Yunghsiang S. Han // Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science. — FOCS '14. — Washington, DC, USA: IEEE Computer Society, 2014. — P. 316–325. — http://dx.doi.org/10.1109/FOCS.2014.41.

112. Ma, J. Divide-and-conquer interpolation for list decoding of Reed-Solomon codes / J. Ma, P. Trifonov, A. Vardy // Proceedings of IEEE International Symposium on Information Theory. — Chicago, USA: IEEE, 2004. — June 27 – July 2. — P. 387.

113. Ma, J. A complexity reducing transformation for the Lee—O'Sullivan interpolation algorithm / J. Ma, A. Vardy // Proceedings of IEEE International Symposium on Information Theory. — 2007. — P. 1986 – 1990.

114. MacMullan, S. J. The capacity of binary channels that use linear codes and decoders / Samuel J. MacMullan, Oliver M. Collins // IEEE Transactions On Information Theory. — 1998. — January. — Vol. 44, no. 1.

115. Massey, J. Variable-length codes and the Fano metric / J. Massey // IEEE Transactions on Information Theory. — 1972. — January. — Vol. 18, no. 1. — P. 196–198.

116. Maucher, J. On the equivalence of generalized concatenated codes and generalized error location codes / Johannes Maucher, V. Zyablov, M. Bossert // IEEE Transactions On Information Theory. — 2000. — March. — Vol. 46, no. 2. — P. 642–649.

117. Miloslavskaya, V. Shortened polar codes / V. Miloslavskaya // IEEE Transactions on Information Theory. — 2015. — Vol. 61, no. 9. — P. 4852–4865.

118. Miloslavskaya, V. Design of polar codes with arbitrary kernels / V. Miloslavskaya, P. Trifonov // Proceedings of IEEE Information Theory Workshop. — 2012. — P. 119–123.

119. Miloslavskaya, V. Sequential decoding of polar codes / V. Miloslavskaya, P. Trifonov // IEEE Communications Letters. — 2014. — Vol. 18, no. 7. — P. 1127–1130.

120. Miloslavskaya, V. Sequential decoding of polar codes with arbitrary binary kernel /

V. Miloslavskaya, P. Trifonov // Proceedings of IEEE Information Theory Workshop. — Hobart, Australia: IEEE, 2014. — P. 377–381.

121. Mondelli, M. Scaling exponent of list decoders with applications to polar codes / Marco Mondelli, S. Hamed Hassani, R.L. Urbanke // IEEE Transactions On Information Theory. — 2015. — September. — Vol. 61, no. 9.
122. Moorthy, H. T. Soft-decision decoding of binary linear block codes based on an iterative search algorithm / Hari T. Moorthy, Shu Lin, Tadao Kasami // IEEE Transactions On Information Theory. — 1997. — May. — Vol. 43, no. 3. — P. 1030–1040.
123. Mori, R. Performance and construction of polar codes on symmetric binary-input memoryless channels / Ryuhei Mori, Toshiyuki Tanaka // Proceedings of IEEE International Symposium on Information Theory. — 2009.
124. Mori, R. Performance of polar codes with the construction using density evolution / Ryuhei Mori, Toshiyuki Tanaka // IEEE Communications Letters. — 2009. — July. — Vol. 13, no. 7.
125. Mori, R. Non-binary polar codes using Reed-Solomon codes and algebraic geometry codes / Ryuhei Mori, Toshiyuki Tanaka // Proceedings of IEEE Information Theory Workshop. — Dublin, 2010. — P. 1–5.
126. Mori, R. Source and channel polarization over finite fields and Reed-Solomon matrix / Ryuhei Mori, Toshiyuki Tanaka // IEEE Transactions on Information Theory. — 2014. — May. — Vol. 60, no. 5. — P. 2720–2736.
127. Mulders, T. On lattice reduction for polynomial matrices / T. Mulders, A. Storjohann // Journal of Symbolic Computation. — 2003. — Vol. 35, no. 4. — P. 377—-401.
128. Network coding for distributed storage systems / Alexandros G. Dimakis, P. Brighten Godfrey, Yunnan Wu et al. // IEEE Transactions on Information Theory. — 2010. — September. — Vol. 56, no. 9.
129. Nielsen, R. R. Decoding Reed-Solomon codes beyond half the minimum distance / R. Refslund Nielsen, T. Hoholdt // Proceedings of the International Conference on Coding Theory and Cryptography. — Mexico: Springer-Verlag, 1998. — P. 221–236.
130. Nielsen, R. R. Decoding Reed-Solomon codes beyond half the minimum distance / R. R. Nielsen, T. Hoholdt // Coding theory, Cryptography and Related Areas. — Springer-Verlag, 2000. — P. 221–236.
131. Niu, K. Stack decoding of polar codes / K. Niu, K. Chen // Electronics Letters. — 2012. — June. — Vol. 48, no. 12. — P. 695–697.
132. Niu, K. Beyond turbo codes: Rate-compatible punctured polar codes / Kai Niu, Kai Chen,

Jiaru Lin // Proceedings of IEEE International Communications Conference. — 2013.

133. O'Keefe, H. Gröbner basis solutions of constrained interpolation problems / Henry O'Keefe, Patrick Fitzpatrick // Linear Algebra and Applications. — 2002. — Vol. 351. — P. 533–551.
134. On the locality of codeword symbols / Parikshit Gopalan, Cheng Huang, Huseyin Simitci, Sergey Yekhanin // IEEE Transactions on Information Theory. — 2012. — November. — Vol. 58, no. 11.
135. On the optimum bit orders with respect to the state complexity of trellis diagrams for binary linear codes / Tadao Kasami, T. Takata, Eiji Fujiwara, Shu Lin // IEEE Transactions On Information Theory. — 1993. — January. — Vol. 39, no. 1.
136. Performance limits and practical decoding of interleaved Reed-Solomon polar concatenated codes / Hessam Mahdavifar, Mostafa El-Khamy, Jungwon Lee, Inyup Kang // IEEE Transactions On Communications. — 2014. — May. — Vol. 62, no. 5. — P. 1406–1417.
137. Pfister, H. Near-optimal finite-length scaling for polar codes over large alphabets / H.D. Pfister, R.L. Urbanke // Proceedings of IEEE International Symposium on Information Theory. — 2016.
138. Plank, J. S. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems / James S. Plank // Software – Practice and Experience. — 1997. — September. — Vol. 27, no. 9. — P. 995–1012.
139. Plank, J. S. Screaming fast Galois field arithmetic using intel SIMD instructions / J. S. Plank, K. M. Greenan, E. L. Miller // Proceedings of 11th Usenix Conference on File and Storage Technologies. — 2013. — February.
140. Poltyrev, G. Bounds on the decoding error probability of binary linear codes via their spectra / Gregory Poltyrev // IEEE Transactions on Information Theory. — 1994. — July. — Vol. 40, no. 4.
141. Polyanskiy, Y. Channel coding rate in the finite block length regime / Yury Polyanskiy, H. Vincent Poor, Sergio Verdu // IEEE Transactions On Information Theory. — 2010. — May. — Vol. 56, no. 5.
142. Proakis, J. G. Digital communications / J. G. Proakis. — McGraw Hill, 1995.
143. R1-1709181: LDPC rate compatible design: Rep. / 3GPP; Executor: Qualcomm: 2017.
144. Rate-dependent analysis of the asymptotic behavior of channel polarization / S. Hamed Hassani, Ryuhei Mori, Toshiyuki Tanaka, R.L. Urbanke // IEEE Transactions On Information Theory. — 2013. — April. — Vol. 59, no. 4.
145. Reed-muller codes achieve capacity on erasure channels / Shrinivas Kudekar, Santhosh Kumar, Marco Mondelli et al. // IEEE Transactions On Information Theory. — 2017. — July. —

Vol. 63, no. 7.

146. Richardson, T. Modern Coding Theory / T. Richardson, R. Urbanke. — Cambridge University Press, 2008.
147. Roder, M. Fast tree-trellis list Viterbi decoding / M. Roder, R. Hamzaoui // IEEE Transactions on Communications. — 2006. — March. — Vol. 54, no. 3. — P. 453–461.
148. Roth, R. Efficient decoding of Reed-Solomon codes beyond half the minimum distance / R. Roth, G. Ruckenstein // IEEE Transactions on Information Theory. — 2000. — Vol. 46, no. 1. — P. 246–257.
149. Roth, R. Encoding and decoding of bch codes using light and short codewords / R.M. Roth, G. Seroussi // IEEE Transactions on Information Theory. — 1988. — May. — Vol. 34, no. 3.
150. Samimi, M. K. 3-d millimeter-wave statistical channel model for 5g wireless system design / Mathew K. Samimi, Theodore S. Rappaport // IEEE Transactions on Microwave Theory and Techniques. — 2016. — July. — Vol. 64, no. 7.
151. Sauer, T. Polynomial interpolation of minimal degree and Gröbner bases / Thomas Sauer // Gröbner Bases and Applications (Proceedings of the International Conference “33 Years of Gröbner Bases”) / Ed. by B. Buchberger, F. Winkler. — Vol. 251 of London Mathematical Society Lecture Notes. — Cambridge University Press, 1998. — P. 483–494.
152. A semi-parallel successive-cancellation decoder for polar codes / Camille Leroux, Alexandre J. Raymond, Gabi Sarkis, W.J. Gross // IEEE Transactions on Signal Processing. — 2013. — January. — Vol. 61, no. 2. — P. 289–299.
153. Sloane, N. J. A. New binary codes / N. J. A. Sloane, S. M. Reddy, C.-L. Chen // IEEE Transactions on Information Theory. — 1972. — July. — Vol. 18, no. 4.
154. Sorger, U. The star trellis of the Golay code / U. Sorger, S. Fedorenko // Proceedings of Seventh International Workshop on Algebraic and Combinatorial Coding Theory. — 2000. — P. 288–292.
155. Study on channel model for frequency spectrum above 6 ghz: Rep.: TR 38.900 V14.2.0 / 3rd Generation Partnership Project (3GPP): 2016. — December.
156. Sudan, M. List decoding: Algorithms and applications / Madhu Sudan // Proceedings of the IFIP International Conference on Theoretical Computer Science. — Vol. 1872 of Lecture Notes in Computer Science. — Sendai, Japan: Springer, 2000. — August. — P. 25–41.
157. Tal, I. List decoding of polar codes / Ido Tal, A. Vardy // Proceedings of IEEE International Symposium on Information Theory. — 2011. — P. 1–5.
158. Tal, I. How to construct polar codes / Ido Tal, Alexander Vardy // IEEE Transactions on Information Theory. — 2013. — October. — Vol. 59, no. 10. — P. 6562–6582.

159. Tal, I. List decoding of polar codes / Ido Tal, A. Vardy // IEEE Transactions On Information Theory. — 2015. — May. — Vol. 61, no. 5. — P. 2213–2226.

160. ten Brink, S. Convergence behavior of iteratively decoded parallel concatenated codes / Stephan ten Brink // IEEE Transactions On Communications. — 2001. — October. — Vol. 49, no. 10.

161. Tm synchronization and channel coding. — Recommended Standard CCSDS 131.0-B-2. — 2011. — August.

162. Trifonov, P. Adaptive data transmission in multi-carrier systems: Ph. D. thesis / Saint-Petersburg State Polytechnic University. — 2005. — May. — in Russian.

163. Trifonov, P. On the relationship of some Reed-Solomon decoding algorithms / P. Trifonov // Proceedings of "Coding Theory Days in Saint-Petersburg" Workshop. — 2008. — P. 83–87.

164. Trifonov, P. Another derivation of Wu list decoding algorithm and interpolation in rational curve fitting / P. Trifonov // Proceedings of IEEE R8 International Conference on Computational Technologies in Electrical and Electronics Engineering. — Irkutsk,Russia: IEEE, 2010. — P. 59–64.

165. Trifonov, P. Efficient interpolation in the Guruswami-Sudan algorithm / P. Trifonov // IEEE Transactions on Information Theory. — 2010. — September. — Vol. 56, no. 9. — P. 4341–4349.

166. Trifonov, P. Efficient design and decoding of polar codes / Peter Trifonov // IEEE Transactions on Communications. — 2012. — November. — Vol. 60, no. 11. — P. 3221 – 3227.

167. Trifonov, P. On the additive complexity of the cyclotomic FFT algorithm / P. Trifonov // Proceedings of IEEE Information Theory Workshop. — Lausanne, Switzerland: IEEE, 2012. — P. 537 – 541.

168. Trifonov, P. Binary successive cancellation decoding of polar codes with Reed-Solomon kernel / P. Trifonov // Proceedings of IEEE International Symposium on Information Theory. — Honolulu, USA: IEEE, 2014. — P. 2972 – 2976.

169. Trifonov, P. Design of polar codes for Rayleigh fading channel / P. Trifonov // Proceedings of International Symposium on Wireless Communication Systems. — Brussels, Belgium: IEEE, 2015.

170. Trifonov, P. Low-complexity implementation of RAID based on Reed-Solomon codes / P. Trifonov // ACM Transactions on Storage. — 2015. — February. — Vol. 11, no. 1. — P. 1:1–1:25.

171. Trifonov, P. Chained polar subcodes / P. Trifonov // Proceedings of 11th International ITG Conference on Systems, Communications and Coding. — Hamburg, Germany: ITG, 2017.

172. Trifonov, P. Star polar subcodes / P. Trifonov // Proceedings of IEEE Wireless Communi-

cations and Networking Conference Workshops. — San-Francisco, USA: IEEE, 2017.

173. Trifonov, P. Chained successive cancellation decoding of the extended Golay code / P. Trifonov // Proceedings of Iran Workshop on Communication and Information Theory. — Tehran, Iran: Sharif University of Technology, 2018.
174. Trifonov, P. Randomized chained polar subcodes / P. Trifonov // Proceedings of IEEE Wireless Communications and Networking Conference Workshops. — Barcelona, Spain: IEEE, 2018.
175. Trifonov, P. Efficient interpolation in Wu list decoding algorithm / Peter Trifonov, Moon Ho Lee // IEEE Transactions on Information Theory. — 2012. — September. — Vol. 58, no. 9. — P. 5963–5971.
176. Trifonov, P. Polar codes with dynamic frozen symbols and their decoding by directed search / P. Trifonov, V. Miloslavskaya // Proceedings of IEEE Information Theory Workshop. — Sevilla, Spain: IEEE, 2013. — September. — P. 1 – 5.
177. Trifonov, P. Twisted polar codes / P. Trifonov, V. Miloslavskaya // Proceedings of International Symposium on Information Theory and Its Applications. — Melbourne, Australia: IEEE, 2014. — P. 456–460.
178. Trifonov, P. Polar subcodes / P. Trifonov, V. Miloslavskaya // IEEE Journal on Selected Areas in Communications. — 2016. — February. — Vol. 34, no. 2. — P. 254–266.
179. Trifonov, P. A randomized construction of polar subcodes / P. Trifonov, G. Trofimiuk // Proceedings of IEEE International Symposium on Information Theory. — Aachen, Germany: IEEE, 2017. — P. 1863–1867.
180. Trofimiuk, G. Block sequential decoding of polar codes / G. Trofimiuk, P. Trifonov // Proceedings of International Symposium on Wireless Communication Systems. — Brussels, Belgium: IEEE, 2015. — P. 326–330.
181. Truong, T.-K. Fast algorithm for computing the roots of error locator polynomials up to degree 11 in Reed-Solomon decoders / T.-K. Truong, J.-H. Jeng, I. S. Reed // IEEE Transactions on Communications. — 2001. — Vol. 49, no. 5. — P. 779–783.
182. Ts 38.212 v2.0.0(2017-12). technical specification group radio access network; nr; multiplexing and channel coding (release 15): Rep. / 3GPP: 2017.
183. Valembois, A. Box and match techniques applied to soft-decision decoding / Antoine Valembois, Marc Fossorier // IEEE Transactions on Information Theory. — 2004. — May. — Vol. 50, no. 5. — P. 796–810.
184. van der Waerden, B. L. Algebra / B. L. van der Waerden. — Springer-Verlag, 1991. — Vol. 1.
185. Vardy, A. Even more efficient bounded-distance decoding of the hexacode, the Golay code,

and the Leech lattice / A. Vardy // IEEE Transactions on Information Theory. — 1995. — September. — Vol. 41, no. 5.

186. Vardy, A. Maximum-likelihood soft decision decoding of BCH codes / A. Vardy, Y. Beery // IEEE Transactions On Information Theory. — 1994. — March. — Vol. 40, no. 2.
187. von zur Gathen, J. Modern Computer Algebra / Joachim von zur Gathen, Juergen Gerhard. — Cambridge University Press, 1999.
188. Wachsmann, U. Multilevel codes: Theoretical concepts and practical design rules / U. Wachsmann, R. F. H. Fischer, J. B. Huber // IEEE Transactions on Information Theory. — 1999. — July. — Vol. 45, no. 5. — P. 1361–1391.
189. Wang, R. A novel puncturing scheme for polar codes / R. Wang, R. Liu // IEEE Communications Letters. — 2014. — October. — Vol. 18, no. 10.
190. Wang, T. Parity-check-concatenated polar codes / Tao Wang, Daiming Qu, Tao Jiang // IEEE Communications Letters. — 2016. — December. — Vol. 20, no. 12.
191. Wang, Y. A fast algorithm for the Fourier transform over finite fields and its VLSI implementation / Y. Wang, X. Zhu // IEEE Journal on Selected Areas in Communications. — 1988. — Vol. 6, no. 3. — P. 572–577.
192. Woerz, T. Iterative decoding for multilevel codes using reliability information / T. Woerz, J. Hagenauer // Proceedings of GLOBECOM'92. — Vol. 3. — 1992. — December. — P. 1779 –1784.
193. Wu, Y. New list decoding algorithms for Reed-Solomon and BCH codes / Y. Wu // IEEE Transactions on Information Theory. — 2008. — August. — Vol. 54, no. 8.
194. Yakuba, N. Multilevel buckets for sequential decoding of polar codes / N. Yakuba, P. Trifonov // Proceedings of IEEE International Symposium on Personal, Indoor, Mobile and Radio Communications. — IEEE, 2015.
195. Zhang, J. Shuffled iterative decoding / Juntan Zhang, Marc P.C. Fossorier // IEEE Transactions on Communications. — 2005. — February. — Vol. 53, no. 2.
196. Zinoviev, V. On the dual distance of BCH codes / V.A. Zinoviev, S.N. Litsyn // Problems of Information Transmission. — 1986. — Vol. 22, no. 4. — P. 272–277.

Список сокращений и условных обозначений

$[n]$ — $\{0, \ldots, n-1\}$

A_m — поляризующее преобразование

$\mathcal{F}$ — множество номеров замороженных символов полярного кода

F_l — $l \times l$ ядро поляризующего преобразования

$\langle Q_i(x,y), 0 \le i \le v \rangle$ = $\{\sum_{i=0}^{v} p_i(x,y) Q_i(x,y) | p_i(x,y) \in \mathbb{F}[x,y]\}$ — идеал многочленов, порожденный $Q_i(x,y)$

$[Q_i(x,y), 0 \le i \le v]$ = $\{\sum_{i=0}^{v} p_i(x) Q_i(x,y) | p_i(x) \in \mathbb{F}[x]\}$ — модуль многочленов, порожденный $Q_i(x,y)$

$Q^{[j_1,j_2]}(x_i, y_i)$ = $\sum\limits_{j_1' \ge j_1} \sum\limits_{j_2' \ge j_2} \binom{j_1'}{j_1} \binom{j_2'}{j_2} q_{j_1' j_2'} x_i^{j_1'-j_1} y_i^{j_2'-j_2}$ — производная Хассе $Q(x,y)$ в точке (x_i, y_i).

$Q(x_i, y_i) = 0^r$ $Q(x,y)$ имеет корень кратности *не менее* r в (x_i, y_i), т.е. $Q^{[j_1,j_2]}(x_i, y_i) = 0, j_1 + j_2 < r$

I_r = $\{Q(x,y) \in \mathbb{F}[x,y] | Q(x_i, y_i) = 0^r, 1 \le i \le n\}$

$M_{r,\rho}$ = $\{Q(x,y) \in I_r | \operatorname{wdeg}_{(0,1)} Q(x,y) < \rho\}$

$\operatorname{LT} Q(x,y)$ — старший член $Q(x,y)$ относительно некоторого упорядочения

$\operatorname{ydeg} Q(x,y) = j$ $\operatorname{LT} Q(x,y) = a x^q y^j$ для некоторого $a \in \mathbb{F}$ и $q \in \mathbb{Z}$.

$|\mathcal{B}|$ — размерность вектора $\mathcal{B}$.

$\Delta(\mathcal{B})$ = $\sum_{j=0}^{s} b_j$, где $\mathcal{B} = (B_0(x,y), \ldots, B_s(x,y))$ — базис Грёбнера некоторого модуля и $\operatorname{LT} B_j(x,y) = a_j x^{b_j} y^j$.

АБГШ — аддитивный белый гауссовский шум

БПА — быстрое преобразование Адамара

БПФ — быстрое преобразование Фурье

БЧХ — (код) Боуза-Чоудхури-Хоквингема

ГПСЧ — генератор псевдослучайных чисел

ГС — (алгоритм) Гурусвами-Судана

ДЗС — динамически замороженный символ

ДПФ — дискретное преобразование Фурье

ИИА — итеративный интерполяционный алгоритм

ЛОПП — логарифмическое отношение правдоподобия

МДР — (код) с максимальным достижимым расстоянием

МП — максимум правдоподобия

ОДЗ	—	ограничение динамического замораживания
ОКК	—	обобщенный каскадный код
ОККПС	—	обобщенный каскадный код с перекрестными связями
ОРП	—	обобщенное разложение Плоткина
ОСШ	—	отношение сигнал/шум
ПИ	—	последовательное исключение
ПО	—	приоритетная очередь
РБЧХ	—	расширенный примитивный (код/ядро) БЧХ в узком смысле
РС	—	(код/ядро) Рида-Соломона
САПИ	—	списочный алгоритм последовательного исключения
СХД	—	система хранения данных
ЭСДТВ	—	экземпляр списочного декодера Тала-Варди