

федеральное государственное автономное образовательное учреждение  
высшего образования  
«Санкт-Петербургский государственный политехнический университет»

На правах рукописи

Милославская Вера Дмитриевна

# **Методы построения и декодирования полярных кодов**

05.13.01 – Системный анализ, управление и обработка информации  
(информатика)

**ДИССЕРТАЦИЯ**

на соискание ученой степени  
кандидата технических наук

Научный руководитель

к. т. н., доц.

Трифонов П.В.



1.5.3.1.	Замена переменных . . . . .	49
1.5.3.2.	Интерполяция . . . . .	51
1.5.4.	Метод Чейза . . . . .	51
1.6.	Выводы по разделу . . . . .	52
<b>Глава 2.</b>	<b>Построение полярных кодов . . . . .</b>	<b>54</b>
2.1.	Полярные подкоды . . . . .	54
2.1.1.	Представление линейных кодов для декодирования методом последовательного исключения . . . . .	54
2.1.2.	Полярные подкоды БЧХ . . . . .	57
2.1.2.1.	Расширенные коды БЧХ . . . . .	57
2.1.2.2.	Предлагаемая конструкция кодов . . . . .	61
2.1.3.	Выводы . . . . .	64
2.2.	Укорочение полярных кодов с ядром Арикана . . . . .	65
2.2.1.	Идея предлагаемого алгоритма . . . . .	66
2.2.2.	Эквивалентные шаблоны для укорочения . . . . .	68
2.2.3.	Алгоритм оптимизации . . . . .	71
2.2.4.	Оценка снизу для вероятности ошибки декодирования . . . . .	75
2.2.5.	Снижение сложности оптимизации . . . . .	80
2.2.6.	Сложность декодирования укороченных кодов . . . . .	82
2.2.7.	Численные результаты . . . . .	83
2.2.8.	Выводы . . . . .	86
2.3.	Построение полярных кодов с произвольным двоичным ядром . . . . .	86
2.3.1.	Вероятность отказа от декодирования информационного символа . . . . .	87
2.3.2.	Точный метод . . . . .	88
2.3.3.	Приближенный метод . . . . .	91
2.3.4.	Численные результаты . . . . .	92
2.3.5.	Выводы . . . . .	95

2.4. Выводы по разделу . . . . .	95
<b>Глава 3. Декодирование полярных кодов . . . . .</b>	<b>97</b>
3.1. Декодирование методом направленного поиска . . . . .	97
3.1.1. Описание алгоритма . . . . .	97
3.1.2. Численные результаты . . . . .	100
3.1.3. Выводы . . . . .	100
3.2. Последовательное декодирование полярных кодов с ядром Ари- кана . . . . .	102
3.2.1. Метрика пути . . . . .	102
3.2.2. Эффективная реализация . . . . .	104
3.2.3. Численные результаты . . . . .	110
3.2.4. Выводы . . . . .	113
3.3. Алгоритм последовательного исключения для полярных кодов с произвольным двоичным ядром . . . . .	113
3.3.1. Точный метод . . . . .	114
3.3.2. Декодирование с помощью порядковых статистик . . . . .	116
3.3.3. Численные результаты . . . . .	117
3.3.4. Выводы . . . . .	117
3.4. Последовательное декодирование полярных кодов с произволь- ным двоичным ядром . . . . .	118
3.4.1. Описание алгоритма декодирования . . . . .	118
3.4.2. Вычисление метрики пути . . . . .	119
3.4.2.1. Функция $R(u_0^i, y_0^{n-1})$ . . . . .	119
3.4.2.2. Функция $\hat{\Omega}(i)$ . . . . .	119
3.4.3. Улучшенный алгоритм декодирования . . . . .	120
3.4.4. Реализация . . . . .	122
3.4.4.1. Промежуточные слои . . . . .	123
3.4.4.2. Последний слой . . . . .	124

3.4.5.	Численные результаты . . . . .	125
3.4.6.	Выводы . . . . .	128
3.5.	Последовательное декодирование кодов Рида-Соломона . . . . .	128
3.5.1.	Алгоритм последовательного исключения . . . . .	129
3.5.2.	Предлагаемый алгоритм . . . . .	129
3.5.3.	Вычисление функции $\hat{\Omega}(i)$ . . . . .	131
3.5.3.1.	Гауссовская аппроксимация . . . . .	131
3.5.3.2.	Упрощенный метод . . . . .	133
3.5.4.	Декодирование двоичного образа кода . . . . .	136
3.5.5.	Численные результаты . . . . .	137
3.5.6.	Выводы . . . . .	140
3.6.	Выводы по разделу . . . . .	141
<b>Глава 4.</b>	<b>Декодирование длинных кодов Рида-Соломона . . . . .</b>	<b>142</b>
4.1.	Послойный алгоритм . . . . .	142
4.1.1.	Построение базиса Грёбнера идеала группы . . . . .	144
4.1.2.	Переход от идеалов групп к идеалу слоя . . . . .	145
4.1.3.	Интерполяция в случае корней переменной кратности . . . . .	147
4.2.	Сходимость рандомизированного алгоритма умножения идеалов . . . . .	147
4.3.	Построение базиса для $M^{(m)}$ . . . . .	149
4.4.	Увеличение кратностей корней . . . . .	150
4.5.	Гибридный алгоритм . . . . .	153
4.5.1.	Описание алгоритма . . . . .	153
4.5.2.	Численные результаты . . . . .	154
4.6.	Комбинаторно-алгебраическое декодирование . . . . .	154
4.6.1.	Описание алгоритма . . . . .	154
4.6.2.	Численные результаты . . . . .	159
4.7.	Выводы по разделу . . . . .	161

<b>Глава 5. Кодирование для систем хранения данных</b> . . . . .	162
5.1. Систематическое кодирование . . . . .	162
5.1.1. Поляризирующее преобразование с $m = 2$ . . . . .	165
5.1.2. Поляризирующее преобразование с произвольным $m$ . . . . .	166
5.2. Быстрое умножение для ядра БЧХ . . . . .	168
5.3. Анализ сложности . . . . .	173
5.3.1. Поляризирующее преобразование с $m = 2$ . . . . .	173
5.3.2. Поляризирующее преобразование с произвольным $m$ . . . . .	174
5.3.3. Ядра Арикана и БЧХ . . . . .	175
5.4. Применение полярных кодов в системах хранения данных . . . . .	176
5.4.1. Предлагаемый метод кодирования . . . . .	176
5.4.2. Балансировка нагрузки для группы дисков . . . . .	178
5.4.3. Балансировка нагрузки между группами дисков . . . . .	182
5.5. Численные результаты . . . . .	185
5.6. Выводы по разделу . . . . .	189
<b>Заключение</b> . . . . .	191
<b>Список используемых обозначений</b> . . . . .	194
<b>Литература</b> . . . . .	195
<b>Приложение А. Акты о внедрении</b> . . . . .	205

# Введение

**Актуальность темы исследования.** Технологии помехоустойчивого кодирования являются неотъемлемой частью современных систем хранения и передачи информации. За годы развития теории помехоустойчивого кодирования было построено большое число разнообразных кодов исправляющих ошибки. Однако их характеристики остаются весьма далекими от теоретических пределов, а вероятность ошибки декодирования, демонстрируемая ими при использовании в системах передачи информации, оказывается значительно хуже принципиально достижимой. Одной из причин этого является невозможность практического использования оптимальных алгоритмов декодирования, сложность которых оказывается чрезмерно высокой.

В 2008 году Е. Ариканом были предложены полярные коды и было показано, что они достигают пропускной способности широкого класса каналов передачи информации. Последнее означает, что для любой сколь угодно малой величины  $p > 0$  существует такое целое число  $m$ , что полярный код длины  $n = 2^m$  со скоростью  $R$ , меньшей пропускной способности канала, обеспечивает вероятность ошибки меньше  $p$ . Конструкция полярных кодов была обобщена С.Б. Корадой, Е. Сасоглу и Р.Л. Урбанке. Кроме того, было показано, что полярные коды относятся к классу обобщенных каскадных кодов, предложенных Э.Л. Блохом и В.В. Зябловым. Полярные коды могут быть также представлены как многоуровневые коды. Последние были предложены Х. Имаи и исследованы У. Ваксманном, Р.Ф. Фишером и Д.Б. Хубером. Конструкции кодов, достигающих пропускной способности канала, рассматривались ранее в работах Дж.Д. Фorni, Э.Л. Блоха, В.В. Зяблова, А.М. Барга, Ж. Земора, Р.М. Рота, В. Скачека и некоторых других исследователей. Отличительной особенностью полярных кодов является простота процедур их построения, кодирования и декодирования, что делает их привлекательными для практического использования.

Однако, эксперименты показывают, что вероятность ошибки декодирова-

ния полярных кодов с практически значимыми параметрами, то есть значениями  $n$  порядка нескольких тысяч, оказывается значительно больше, чем у кодов с малой плотностью проверок на четность (МППЧ) и турбо-кодов с аналогичными параметрами. Это обусловлено малым минимальным расстоянием полярных кодов и субоптимальностью алгоритма последовательного исключения, используемого для их декодирования. И. Талом и А. Варди был предложен алгоритм списочного декодирования, построенный на базе алгоритма последовательного исключения и обеспечивающий декодирование полярных кодов почти по максимуму правдоподобия. Также ими была предложена конструкция полярных кодов с контрольной суммой, демонстрирующая существенно меньшую вероятность ошибки декодирования по сравнению с классическими полярными кодами. К. Нию и К. Чень предложили обобщение стекового алгоритма К.Ш. Зигангирова. При меньшей вычислительной сложности этот алгоритм обеспечивает ту же вероятность ошибки декодирования, что и списочный алгоритм Тала-Варди. Тем не менее, сложность декодирования полярных кодов остается выше, чем существующих аналогов. Указанные проблемы препятствуют широкому практическому применению полярных кодов.

Одним из наиболее широко используемых классов корректирующих кодов являются коды Рида-Соломона. При этом остается актуальной задача построения эффективных алгоритмов их мягкого декодирования. Следует отметить, что сложность существующих алгоритмов мягкого декодирования кодов Рида-Соломона, таких как метод Кёттера-Варди, является достаточно высокой. Это приводит к тому, что системы передачи и хранения информации, использующие коды Рида-Соломона, демонстрируют энергетический проигрыш порядка 3 дБ по сравнению с теоретическим пределом. Наиболее трудоемким шагом алгоритма Кёттера-Варди является построение полинома от двух переменных, имеющего корни различной кратности. Вопрос построения быстрых алгоритмов, реализующих этот шаг, исследовался Р. Кёттером, Р.Р. Нильсоном, К. Ли и М.Е. О'Салливаном. Следует отметить, что метод Кёттера-Варди не обеспечивает де-



кодирование кодов Рида-Соломона по максимуму правдоподобия, и открытой остается задача построения алгоритмов декодирования с большей корректирующей способностью.

**Цель диссертационной работы** состоит в создании методов кодирования и декодирования информации, основанных на теории полярных кодов, обеспечивающих меньшую вероятность ошибки и меньшую сложность декодирования по сравнению с существующими аналогами с практически значимыми параметрами. Для достижения поставленной цели необходимо решить следующие **задачи**:

1. Разработать быстрый алгоритм мягкого декодирования двоичных полярных кодов, обеспечивающий меньшую вероятность ошибки декодирования по сравнению с методом последовательного исключения.
2. Разработать конструкции кодов, декодирование которых может быть выполнено с помощью предложенного быстрого алгоритма, обладающих при этом большим минимальным расстоянием по сравнению с полярными кодами с ядром Арикана.
3. Разработать метод укорочения полярных кодов.
4. Разработать быстрый алгоритм мягкого декодирования кодов Рида-Соломона.
5. Разработать высокопроизводительный метод кодирования информации для отказоустойчивых систем хранения данных.

**Научная новизна:**

1. Предложена конструкция полярных подкодов Боуза-Чоудхури-Хоквингема с минимальным расстоянием большим, чем у полярных кодов с ядром Арикана с аналогичными параметрами.
2. Разработан новый метод построения укороченных полярных кодов.

3. Предложен новый метод последовательного декодирования полярных кодов, основанный на использовании оценок максимума апостериорных вероятностей кодовых слов.
4. Разработан новый алгоритм декодирования кодов Рида-Соломона, основанный на методе последовательного декодирования полярных кодов.
5. Предложен новый алгоритм, реализующий этап двумерной интерполяции в методе Кёттера-Варди декодирования кодов Рида-Соломона.
6. Разработан новый высокопроизводительный метод кодирования информации укороченными полярными подкодами (в частности, полярными кодами) для отказоустойчивых систем хранения данных.

**Теоретическая и практическая значимость работы.** В работе представлен набор методов построения и декодирования полярных кодов, а также кодов Рида-Соломона. Эти методы могут найти свое применение в современных и перспективных системах передачи информации.

Предложен метод построения подкодов расширенных кодов Боуза-Чоудхури-Хоквингема (полярных подкодов БЧХ), обеспечивающих меньшую вероятность ошибки при декодировании с помощью стекового и списочного алгоритмов последовательного исключения, чем известные классы полярных кодов, в частности, полярные коды с ядром Арикана. Предложенный метод построения укороченных полярных кодов позволяет получить коды произвольной длины, демонстрирующие высокую корректирующую способность при использовании метода последовательного исключения. Полярные коды с произвольным двоичным ядром, построенные для двоичного стирающего канала с помощью предложенного алгоритма, обеспечивают малую вероятность ошибки декодирования и в Гауссовском канале.

В отличие от классического метода последовательного декодирования и алгоритма последовательного исключения, предложенный метод декодирования

полярных кодов оперирует оценками максимума апостериорных вероятностей кодовых слов. Предложенный метод декодирования полярных кодов имеет существенно меньшую вычислительную сложность по сравнению со стековым и списочным алгоритмами последовательного исключения, при незначительном увеличении вероятности ошибки декодирования. Основанный на нем алгоритм декодирования кодов Рида-Соломона в случае коротких кодов обеспечивает меньшую вероятность ошибки декодирования, чем метод Кёттера-Варди мягкого декодирования кодов Рида-Соломона. Разработанный алгоритм двумерной интерполяции позволяет снизить вычислительную сложность метода Кёттера-Варди.

Одной из возможных сфер применения предложенных методов кодирования и декодирования являются системы мобильной связи. При этом применение предлагаемого метода декодирования полярных кодов позволяет существенно снизить энергопотребление приемного оборудования, в то время как использование полярных подкодов БЧХ позволяет расширить зону уверенного приема по сравнению с аналогичными системами, использующими коды МППЧ.

Разработанный метод кодирования информации укороченными полярными подкодами (в частности, полярными кодами) был использован компанией ООО «Санкт-Петербургский Центр Разработок ЕМС» при разработке высокопроизводительных отказоустойчивых систем хранения данных, что подтверждается актом о внедрении. Предложенный метод превосходит по производительности существующие аналоги, а также предотвращает неравномерный износ носителей информации.

**Положения, выносимые на защиту:**

1. Метод построения кодов, являющихся подкодами расширенных кодов Боуза-Чоудхури-Хоквингема и обеспечивающих меньшую вероятность ошибки декодирования с помощью стекового алгоритма последовательного исключения, чем полярные коды.
2. Метод построения укороченных полярных кодов.

3. Метод последовательного декодирования двоичных полярных кодов и основанный на нем алгоритм декодирования коротких кодов Рида-Соломона.
4. Быстрый алгоритм, реализующий этап двумерной интерполяции в методе Кёттера-Варди декодирования кодов Рида-Соломона.
5. Метод кодирования информации укороченными полярными подкодами (в частности, полярными кодами) для отказоустойчивых систем хранения данных.

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

1. IEEE R8 International Conference on Computational Technologies in Electrical and Electronics Engineering, 2010.
2. 8-th IEEE International Symposium on Wireless Communication Systems, 2011.
3. IEEE Information Theory Workshop, 2012.
4. International Workshop on Algebraic and Combinatorial Coding Theory, 2012.
5. IEEE Information Theory Workshop, 2013.
6. International Symposium on Information Theory and Applications, 2014.
7. IEEE Information Theory Workshop, 2014.

Кроме того, результаты были представлены на семинарах в институте проблем передачи информации им. А.А. Харкевича Российской академии наук (руководитель Л.А. Бассальго) и Санкт-Петербургском государственном университете аэрокосмического приборостроения (руководитель Е.А. Крук).

Предлагаемые алгоритмы были реализованы на языке программирования C++. Выполнено сопоставление результатов статистического моделирования с известными опубликованными данными.

**Публикации.** Материалы диссертации опубликованы в 9 печатных работах [7, 53–59, 78], из них 2 статьи в рецензируемых журналах [7, 57], включенных в список ВАК, и 7 статей в сборниках трудов конференций.

Получено свидетельство о государственной регистрации программы для ЭВМ [6]. Подана заявка на патент [8].

**Личный вклад автора.** Содержание диссертации и основные положения, выносимые на защиту, отражают персональный вклад автора в опубликованные работы. Подготовка к публикации полученных результатов проводилась совместно с соавторами, причем вклад диссертанта был определяющим. Все представленные в диссертации результаты получены лично автором.

**Структура и объем диссертации.** Диссертация состоит из введения, пяти разделов, заключения и библиографии. Общий объем диссертации 206 страниц, из них 187 страниц текста, включая 67 рисунков. Библиография включает 83 наименования на 10 страницах.

В разделе 1 дается описание полярных кодов и некоторых существующих методов их декодирования. Кроме того, рассматривается метод Кёттера-Варди декодирования кодов Рида-Соломона.

В разделе 2 представлены новые конструкции кодов. Вводится представление линейных блоковых кодов в форме, обеспечивающей возможность их декодирования методом последовательного исключения и его аналогами, и основанная на этом представлении конструкция полярных подкодов БЧХ. Дается описание предлагаемого метода построения укороченных полярных кодов с ядром Арикана. Кроме того, представлен алгоритм построения двоичных полярных кодов с произвольным ядром.

Раздел 3 посвящен новым методам декодирования полярных кодов. Представлен метод последовательного декодирования двоичных полярных кодов и

основанный на нем алгоритм декодирования коротких кодов Рида-Соломона, использующий их представление, предложенное в разделе 2.

В разделе 4 рассматривается задача быстрого декодирования длинных кодов Рида-Соломона. В частности, представлен новый быстрый алгоритм, реализующий этап двумерной интерполяции в методе Кёттера-Варди.

Раздел 5 посвящен применению полярных кодов в отказоустойчивых системах хранения данных. Представлены новые методы быстрого кодирования и балансировки нагрузки.

В заключении кратко перечислены основные результаты, полученные в диссертационной работе.

Приложение А содержит копии актов о внедрении.

# Глава 1

## Полярные коды и коды Рида-Соломона

### 1.1. Полярные коды

Полярные коды, предложенные Ариканом в [12], достигают пропускной способности любого симметричного по выходу канала без памяти с двоичным входом (СВКБПДВ). Конструкции кодов, достигающих пропускной способности канала, рассматривались ранее в таких работах, как [2, 14, 30, 36, 50, 68, 83]. Отличительной особенностью полярных кодов является простота процедур их построения, кодирования и декодирования, что делает их привлекательными для практического использования.

В основе конструкции полярных кодов лежит линейное преобразование, задаваемое матрицей  $M^{\otimes m}$ , где операция  $\otimes m$  обозначает  $m$ -кратное Кронекеровское произведение матрицы с собой, и  $l \times l$  ядро поляризации  $M$  является двоичной обратимой матрицей такой, что никакая перестановка ее столбцов не приводит к получению верхнетреугольной матрицы. В [12] рассматривались только полярные коды с  $2 \times 2$  ядром. Обобщение конструкции полярных кодов на случай двоичных ядер произвольной размерности было предложено в [44].

В дальнейшем последовательность вида  $(a_i, \dots, a_j)$  будет обозначаться как  $a_i^j$ . Для заданных  $l$  и  $m$  построим перестановочную матрицу  $\Lambda^{(l,m)}$  такую, что для любого  $t_0^{m-1} \in \{0, \dots, l-1\}^m$  выполняется  $\Lambda_{t,t'}^{(l,m)} = 1$ , где  $t = \sum_{i=0}^{m-1} t_i l^i$  и  $t' = \sum_{i=0}^{m-1} t_{m-1-i} l^i$ .

**Определение 1.**  $(n = l^m, k)$  полярным кодом с  $l \times l$  ядром  $M$  и с множеством замороженных символов  $\mathcal{F} \subset \{0, \dots, n-1\}$ ,  $|\mathcal{F}| = n - k$ , называется линейный блочный код с множеством кодовых слов  $\{u_0^{n-1} G_n | u_0^{n-1} \in \{0, 1\}^n, \forall i \in \mathcal{F} u_i = 0\}$ , где  $G_n = \Lambda^{(l,m)} M^{\otimes m}$  — матрица поляризующего преобразования.

Рассмотрим СВКБПДВ  $W: \mathbb{F}_2 \rightarrow \mathcal{Y}$ , на вход которого с равной вероятностью подаются 0 или 1, а на выходе наблюдаются символы из множества  $\mathcal{Y}$ . Канал передачи данных  $W$  с переходными вероятностями  $W(y|0)$  и  $W(y|1)$ ,  $y \in \mathcal{Y}$ , характеризуется пропускной способностью

$$I(W) = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathbb{F}_2} \frac{1}{2} W(y|x) \log \frac{2W(y|x)}{W(y|0) + W(y|1)}$$

и параметром Бхаттачарьи

$$Z(W) = \sum_{y \in \mathcal{Y}} \sqrt{W(y|0)W(y|1)}.$$

Пропускная способность  $I(W)$  является наибольшей возможной скоростью, при которой возможна надежная передача данных по каналу  $W$ , где под скоростью понимается отношение числа информационных символов к общему числу передаваемых символов. Параметр Бхаттачарьи  $Z(W)$  является верхней границей для вероятности принятия неправильного решения относительно значения символа, полученного на выходе канала  $W$ .

Предполагается, что символы кодового слова  $c_0^{n-1} = u_0^{n-1} G_n$  передаются независимо по каналу  $W$ , это можно рассматривать как передачу кодового слова  $c_0^{n-1}$  по  $n$  независимым копиям канала  $W$ . Если объединить данные копии, то получим канал с переходной вероятностью  $W_{G_n}(y_0^{n-1}|u_0^{n-1}) = \prod_{j=0}^{n-1} W(y_j|c_j)$ . Как показано в [12], полученный канал можно представить в виде объединения  $n$  подканалов  $W_{G_n}^{(i)}: \mathbb{F}_2 \rightarrow \mathcal{Y}^n \times \mathbb{F}_2^i$ ,  $0 \leq i < n$ , которым соответствуют переходные вероятности

$$W_{G_n}^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i) = \sum_{u_{i+1}^{n-1} \in \mathbb{F}_2^{n-i-1}} \frac{1}{2^{n-1}} W_{G_n}(y_0^{n-1}|u_0^{n-1}). \quad (1.1)$$

В [44] было показано, что для матрицы  $G_n$ , построенной на базе  $l \times l$  ядра поляризации  $M$ , и для любого  $\delta > 0$  справедливо

$$\lim_{n \rightarrow \infty} \frac{\left| \{i \in \{0, \dots, n-1\} : I(W_{G_n}^{(i)}) \in (\delta, 1-\delta)\} \right|}{n} = 0,$$



$$\lim_{n \rightarrow \infty} \frac{\left| \{i \in \{0, \dots, n-1\} : Z(W_{G_n}^{(i)}) \in (\delta, 1-\delta)\} \right|}{n} = 0.$$

На практике для построения кодов конечной длины удобно упорядочивать подканалы поляризующего преобразования по их вероятностям ошибки  $P_i = P(E_i | \bar{E}_0, \dots, \bar{E}_{i-1})$ , где  $E_j$  и  $\bar{E}_j$  — события, соответствующие принятию неправильного и правильного решения относительно значения символа  $u_j$ , соответственно [73].

**Определение 2.** *Классическим полярным кодом будем называть полярный код, порождаемый строками с индексами  $i$  матрицы поляризующего преобразования  $G_n$ , которым соответствуют подканалы  $W_{G_n}^{(i)}$  с наименьшими вероятностями ошибки  $P_i$ .*

Рассмотрим равномерно распределенную случайную величину  $W_n$ , которой соответствует множество значений  $\{W_{G_n}^{(i)}\}_{i=0}^{n-1}$ . Корректирующая способность  $(n, k)$  классического полярного кода определяется экспонентой поляризации используемого ядра.  $l \times l$  ядро  $M$  характеризуется экспонентой поляризации  $\Xi(M)$ , если для любого  $\beta < \Xi(M)$  выполняется условие  $\lim_{n \rightarrow \infty} \Pr[Z(W_n) \leq 2^{-n^\beta}] = I(W)$ , и для любого  $\beta > \Xi(M)$  выполняется условие  $\lim_{n \rightarrow \infty} \Pr[Z(W_n) \geq 2^{-n^\beta}] = 1$ , при  $0 < I(W) < 1$ .

Как показано в [44], для заданного ядра  $M$  экспонента поляризации  $\Xi(M)$  может быть вычислена следующим образом

$$\Xi(M) = \frac{1}{l} \sum_{i=0}^{l-1} \log_l D_i,$$

где  $i$ -ое частичное расстояние  $D_i$  равно наименьшему расстоянию Хэмминга между  $i$ -ой строкой матрицы  $M$  и кодовым словом кода, порождаемого  $i+1, \dots, l-1$  строками матрицы  $M$ . При этом  $D_{l-1}$  полагается равным весу  $(l-1)$ -ой строки матрицы  $M$ . Эти частичные расстояния удовлетворяют следующему условию

$$Z(W)^{D_i} \leq Z(W_M^{(i)}) \leq 2^{l-i} Z(W)^{D_i}.$$

### 1.1.1. Ядро Арикана

Первоначально полярные коды были предложены [12] только для случая ядра поляризации

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \quad (1.2)$$

которое в дальнейшем будет называться ядром Арикана. Экспонента поляризации как ядра Арикана  $A$ , так и  $A^{\otimes m}$  равна  $1/2$ . Также в [12] показано, что в случае ядра Арикана параметр Бхаттачарьи подканалов  $W_{G_n}^{(i)}$  удовлетворяет условиям

$$Z(W_{G_{2n}}^{(2i)}) \leq 2Z(W_{G_n}^{(i)}) - Z(W_{G_n}^{(i)})^2, \quad (1.3)$$

$$Z(W_{G_{2n}}^{(2i+1)}) = Z(W_{G_n}^{(i)})^2. \quad (1.4)$$

В случае двоичного стирающего канала получаем точные равенства для вероятностей стирания  $p(W_{G_n}^{(i)})$  в битовых подканалах  $W_{G_n}^{(i)}$

$$p(W_{G_{2n}}^{(2i)}) = 2p(W_{G_n}^{(i)}) - p(W_{G_n}^{(i)})^2, \quad (1.5)$$

$$p(W_{G_{2n}}^{(2i+1)}) = p(W_{G_n}^{(i)})^2. \quad (1.6)$$

Полярные коды, построенные на базе ядра Арикана, получили широкое распространение, поскольку для них существуют эффективные алгоритмы построения, кодирования и декодирования, обладающие малой вычислительной сложностью. Минимальное расстояние  $d$  полярного кода с ядром Арикана длины  $n$  равно наименьшему весу строки матрицы  $G_n$ , соответствующей информационному символу, то есть  $d = \min_{i \notin \mathcal{F}} \text{wt}((G_n)_i)$ . В [35] было показано, что в случае классического полярного кода  $d$  пропорционально  $\sqrt{n}$ .

### 1.1.2. Ядро БЧХ

В [44] было показано, что ядра поляризации размерности  $l > 2$  могут обладать скоростью поляризации большей, чем ядро Арикана. Для построения ядер

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	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	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	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1	1	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	1	1	1	0	0	0	0	0
1	1	1	1	0	1	1	0	0	1	0	1	0	0	0	0
1	0	1	1	1	0	1	1	0	0	1	0	1	0	0	0
1	0	0	1	1	1	0	1	1	0	0	1	0	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Рис. 1.1.  $16 \times 16$  ядро БЧХ

большой размерности в [44] было предложено использовать укороченные коды Боуза-Чоудхури-Хоквингема (БЧХ).

В данном разделе рассматривается метод построения  $l \times l$  нижнетреугольного ядра поляризации  $B$ , подматрицами которого являются порождающие матрицы расширенных кодов БЧХ длины  $l = 2^\gamma$ . Кодом БЧХ над полем  $\mathbb{F}_2$  длины  $2^\gamma - 1$  с конструктивным расстоянием  $\delta$  называется двоичный циклический код наибольшей размерности, множество корней порождающего многочлена которого включает  $\alpha^b, \alpha^{b+1}, \dots, \alpha^{b+\delta-2}$ , где  $\alpha \in \mathbb{F}_{2^\gamma}$  — элемент порядка  $2^\gamma - 1$ ,  $b \in \{0, \dots, 2^\gamma - 2\}$ .

Рассмотрим множество вложенных расширенных кодов БЧХ с порождающими полиномами  $g_0(x), \dots, g_\tau(x)$  такими, что  $g_0(x) = 1$ ,  $g_{t+1}(x) = w_t(x)g_t(x)$ , где  $w_t(x)$  — минимальный полином соответствующего элемента из поля  $\mathbb{F}_{2^\gamma}$ ,  $g_t(x)$  имеет корни  $\alpha, \alpha^2, \dots, \alpha^{\delta_t-1}$ ,  $\delta_t < \delta_{t+1}$ ,  $\delta_\tau = 2^\gamma - 1$ , где  $\delta_t$  — конструктив-

ное минимальное расстояние соответствующего кода длины  $(2^\gamma - 1)$ . Пусть  $h$ -ая строка матрицы  $B$  (за исключением 0-го элемента) является вектором коэффициентов следующего полинома

$$\pi_h(x) = x^j g_t(x), \quad (1.7)$$

где  $j = h - \deg g_t(x) - 1$ ,  $1 \leq h < l$  и  $t$  удовлетворяет условию  $\deg g_t(x) \leq h - 1 < \deg g_{t+1}(x)$ . 0-ая строка матрицы  $B$  содержит единицу на 0-ой позиции и нули на прочих позициях. 0-ой столбец матрицы содержит биты проверки на четность для соответствующих строк, исключением является 0-ой элемент, который равен 1. Последовательность частичных минимальных расстояний такой матрицы  $B$  равна последовательности минимальных расстояний соответствующих расширенных кодов БЧХ, что упрощает вычисление скорости поляризации. Для ядра БЧХ  $16 \times 16$ ,  $32 \times 32$  и  $64 \times 64$  скорости поляризации равны 0.51828, 0.53656 и 0.564271, соответственно. На Рис. 1.1 приведен пример ядра БЧХ для  $l = 16$ .

### 1.1.3. Полярные коды как обобщенные каскадные коды

Обобщенные каскадные коды (ОКК) были предложены в [1]. Рассмотрим ОКК, задаваемый семейством двоичных вложенных внутренних  $(v, v - i, \delta_i)$  кодов  $\mathbb{C}^i$  и двоичных внешних  $(n, k_i, d_i)$  кодов  $\mathcal{C}^i$ ,  $0 \leq i < v$ . Такой ОКК характеризуется длиной  $N = vn$ , размерностью  $K = \sum_{i=0}^{v-1} k_i$  и минимальным расстоянием  $D = \min_{0 \leq i < v} d_i \delta_i$ .

При кодировании ОКК сначала выполняется разбиение информационной последовательности  $u_{0..K-1}$  на блоки  $a_{i,0..k_i-1}$ ,  $0 \leq i < v$ .  $i$ -ый блок кодируется внешним кодом  $\mathcal{C}^i$ , результатом чего является кодовое слово  $b_{i,0..v-1}$ ,  $0 \leq i < v$ . Затем последовательности  $b_{0..v-1,j}$ , состоящие из  $j$ -ых символов кодовых слов внешних кодов, кодируются внутренним кодом  $\mathbb{C}^0$ , получаемые кодовые слова  $(c_{j,0}, \dots, c_{j,v-1})$ ,  $0 \leq j < n$ , составляют кодовое слово ОКК.

В статье [76] было показано, что рекурсивная структура полярных кодов

позволяет представлять их в виде ОКК. Порождающая матрица  $(N = l^m, K)$  полярного кода с  $l \times l$  ядром  $M$  состоит из строк  $N \times N$  матрицы  $M^{\otimes m}$ , соответствующих незамороженным битовым подканалам. Представление матрицы  $M^{\otimes m}$  в виде  $M^{\otimes s} \otimes M^{\otimes(m-s)}$  позволяет выделить внутренние  $(v = l^s, v - i)$  коды  $C_i$ , порождаемые последними  $(v - i)$  строками матрицы  $M^{\otimes s}$ , и внешние  $(n = l^{m-s}, k_i)$  коды  $C_i$ , порождаемые  $k_i$  строками матрицы  $M^{\otimes(m-s)}$ , номера которых определяются множеством замороженных символов  $\mathcal{F}$ ,  $\sum_0^{v-1} k_i = K$ ,  $0 \leq i < v$ . Заметим, что внутренние коды  $C_i$  и внешние коды  $C_i$  также являются полярными,  $0 \leq i < v$ .

#### 1.1.4. Коды произвольной длины

Существующие методы построения полярных кодов произвольной длины могут быть разделены на две группы. Первая группа предполагает использование каскадных кодов с внутренними или внешними полярными кодами [29, 63, 69, 79]. Вторая группа включает конструкции полярных кодов, основанные на выкалывании. При выкалывании  $s$  символов, позиции которых задаются некоторым шаблоном  $T$ , из  $(N, K, D)$  линейного кода получается  $(n = N - s, k \leq K, d \leq D)$  линейный код, где шаблон  $T$  является двоичным вектором длины  $N$  и веса  $s$ . Выкалывание состоит в исключении символов  $c_i$  кодового слова  $c_0^{N-1}$ , для которых  $T_i = 1$ ,  $0 \leq i < N$ . Вероятность ошибки декодирования полученного  $(n, k, d)$  кода зависит от выбора шаблона  $T$ . Основанный на выкалывании метод построения  $L \times L$  поляризующей матрицы из  $2^m \times 2^m$  матрицы Арикана  $A^{\otimes m}$  и множества замороженных битовых подканалов был предложен в [70], где  $2^{m-1} \leq L \leq 2^m$ . При построении такой  $L \times L$  матрицы ставится задача максимизации ее экспоненты поляризации. При этом ввиду большого числа шаблонов выкалывания для кода длины  $2^m$ , исходный полярный код рассматривается как обобщенный каскадный код с внутренним кодом длины  $2^\mu$  и внешними кодами длины  $2^{m-\mu}$ , и осуществляется перебор шаблонов выкалывания для внутренне-

го кода. Получаемый в результате код длины  $L$  является выколотым полярным кодом. Оптимизация шаблонов для выкалывания также рассматривалась в [29], где был предложен алгоритм, основанный на анализе блокирующих множеств соответствующего графа Таннера. Выколотыми полагаются символы кодового слова, входящие в наибольшее число блокирующих множеств. Этот подход основан на результатах практических экспериментов, показывающих, что в случае двоичного стирающего канала вероятность восстановления оказывается выше для тех символов, которые входят в меньшее число блокирующих множеств. Было показано, что корректирующая способность кодов, получаемых с помощью данных субоптимальных методов, выше, чем в случае случайного выкалывания, однако, она существенно ниже, чем у кодов МППЧ и турбо-кодов.

Одним из стандартных методов построения кода произвольной длины  $n$  из заданного кода длины  $N \geq n$  является укорочение. Укорочение произвольного линейного блочного кода  $C$  длины  $N$  на  $s = \text{wt}(T)$  символов состоит в выборе всех кодовых слов  $c_0^{N-1} \in C$  таких, что  $c_i = 0$  для  $i \in \text{supp}(T)$  и исключении из этих кодовых слов символов  $c_i$  для  $i \in \text{supp}(T)$ , полученные таким образом вектора составляют  $(n = N - s, k \geq K - s, d \geq D)$  линейный код. Как и в случае выкалывания, использование различных шаблонов  $T$  приводит к построению кодов, вероятности ошибки декодирования которых существенно различаются. При построении кода длины  $n$  из кода длины  $N$  число возможных шаблонов  $T$  равно  $\binom{N}{N-n}$ .

## 1.2. Систематическое кодирование полярных кодов с ядром

### Арикана

Для случая полярных кодов, построенных на базе  $2 \times 2$  ядра Арикана, в статье [13] был предложен алгоритм систематического кодирования. Идея этого алгоритма состоит в кодировании через декодирование:  $k$  символов кодового слова равны информационным символам, значения остальных  $n - k$  символов

кодированного слова полагаются стертыми и восстанавливаются с помощью декодера. Алгоритм основан на следующих свойствах матрицы  $A^{\otimes m}$ , из  $k$  строк которой состоит порождающая матрица полярного кода:

1. Матрицы  $A^{\otimes m}$  связаны рекурсивным соотношением

$$A^{\otimes(m+1)} = \begin{pmatrix} A^{\otimes m} & 0_{2^m} \\ A^{\otimes m} & A^{\otimes m} \end{pmatrix}, \quad (1.8)$$

где  $0_{2^m}$  —  $2^m \times 2^m$  нулевая матрица.

2.  $A^{\otimes m}$  является нижнетреугольной матрицей с единичной диагональю. Она всегда обратима.
3. Любая подматрица  $A_{\mathcal{D},\mathcal{D}}^{\otimes m}$  матрицы  $A^{\otimes m}$ , состоящая из строк и столбцов с индексами из множества  $\mathcal{D} \subset \{0, \dots, n-1\}$ , также является нижнетреугольной с единичной диагональю, и потому обратима.

При вычислениях используется рекурсивная структура порождающей матрицы, задаваемая формулой (1.8). Предполагается, что множество позиций информационных символов в кодированном слове совпадает с множеством  $\mathcal{N} = \{0, \dots, n-1\} \setminus \mathcal{F}$ ,  $|\mathcal{N}| = k$ , где  $\mathcal{F}$  — множество индексов замороженных символов во входной последовательности. При систематическом кодировании выполняется поиск решения системы линейных уравнений

$$c_0^{n-1} = u_0^{n-1} A^{\otimes m}. \quad (1.9)$$

В данной системе неизвестными являются элементы последовательностей  $c_{0,\mathcal{F}}^{n-1}$  и  $u_{0,\mathcal{N}}^{n-1}$ ,  $u_{a,D}^b$  — подпоследовательность последовательности  $u_a^b$ , состоящая из элементов  $u_i$ ,  $i \in D$ . Благодаря рекурсивной структуре матрицы  $A^{\otimes m}$ , систему (1.9) можно представить в следующем виде

$$c_{n/2}^{n-1} = u_{n/2}^{n-1} A^{\otimes(m-1)}, \quad c_0^{n/2-1} - c_{n/2}^{n-1} = u_0^{n/2-1} A^{\otimes(m-1)}.$$

```

POLARSYSENCODE( $c_0^{n-1}, n, \mathcal{F}$ )
1  if ( $n = 1$ )
2    then if  $0 \in \mathcal{F}$ 
3      then  $c_0 \leftarrow 0$ 
4    else POLARSYSENCODE( $c_{n/2}^{n-1}, n/2, \{i - n/2 | i \in \mathcal{F}, i \geq n/2\}$ )
5       $\mathcal{N} \leftarrow \{0, \dots, n-1\} \setminus \mathcal{F}$ 
6       $v_{0,\mathcal{N}}^{n/2-1} \leftarrow c_{0,\mathcal{N}}^{n/2-1} - c_{n/2,\mathcal{N}}^{n-1}$ 
7      POLARSYSENCODE( $v_0^{n/2-1}, n/2, \{i | i \in \mathcal{F}, i < n/2\}$ )
8       $c_{0,\mathcal{F}}^{n/2-1} \leftarrow v_{0,\mathcal{F}}^{n/2-1} + c_{n/2,\mathcal{F}}^{n-1}$ 

```

Рис. 1.2. Алгоритм Арикана для систематического кодирования полярных кодов

Таким образом, задача поиска решения системы (1.9) может быть сведена к поиску решений двух систем в вдвое меньшей размерности.

Алгоритм систематического кодирования для полярных кодов с ядром Арикана представлен на Рис. 1.2. На вход данному рекурсивному алгоритму подается кодовое слово  $c_0^{n-1}$  с заданными значениями информационных символов  $c_{0,\mathcal{N}}^{n-1}$ , длина полярного кода  $n$ , множество замороженных символов  $\mathcal{F}$ . Результатом работы алгоритма являются вычисленные значения проверочных символов  $c_{0,\mathcal{F}}^{n-1}$ . На шагах 4 и 7 осуществляется переход к задаче систематического кодирования меньшей размерности.

Сложность алгоритма систематического кодирования Арикана можно выразить как  $N_n = \frac{1}{2}n \log n$  операций сложения для элементов  $\mathbb{F}_2$ .

## 1.3. Декодирование полярных кодов

### 1.3.1. Алгоритм последовательного исключения

#### 1.3.1.1. Описание алгоритма

Для принятой последовательности  $y_0^{n-1}$  задача декодирования входной последовательности в двоичном полярном коде длины  $n = l^m$  с  $l \times l$  ядром  $M$  состо-



ит в поиске  $\arg \max_{u_0^{n-1}: u_{0,\mathcal{F}}^{n-1}=\mathbf{0}} P(u_0^{n-1}|y_0^{n-1})$ . Субоптимальное решение  $\hat{u}_0^{n-1}$  задачи декодирования может быть найдено с помощью алгоритма последовательного исключения, предполагающего последовательное принятие решений относительно значений символов входной последовательности. На  $i$ -ой фазе алгоритма принимается решение

$$\hat{u}_i = \begin{cases} \arg \max_{u_i \in \{0,1\}} P(\hat{u}_0^{i-1}, u_i | y_0^{n-1}), & i \notin \mathcal{F} \\ 0, & i \in \mathcal{F}. \end{cases} \quad (1.10)$$

Для заданного пути  $u_0^i$  вероятность  $P(u_0^i | y_0^{n-1})$  может быть рекурсивно вычислена согласно выражению

$$P(u_0^{ls+t} | y_0^{n-1}) = \sum_{u_{ls+t+1}^{l(s+1)-1}} \prod_{j=0}^{l-1} \pi_j, \quad (1.11)$$

где  $\pi_j = P\left(\theta_M \left[ u_0^{(s+1)l-1}, j \right] | y_j^{(j+1)\frac{n}{l}-1} \right)$ ,  $\theta_M \left[ u_0^{l(s+1)-1}, j \right]_r = \left( u_{lr}^{l(r+1)-1} M \right)_j$ ,  $0 \leq t < l$ ,  $0 \leq r \leq s$ ,  $P(u_r | y_r) = \frac{P(y_r | u_r) P(u_r)}{P(y_r)}$ , и  $P(y_r | u_r)$  — переходная вероятность для  $r$ -го символа.

Вероятность ошибки декодирования полярного кода методом последовательного исключения выражается как

$$P^{(e)} = 1 - \prod_{i \notin \mathcal{F}} (1 - P_i) \quad (1.12)$$

где  $P_i$  — вероятность ошибки в подканале  $W_{G_n}^{(i)}$ , задаваемом выражением (1.1), которая также может рассматриваться как вероятность принятия ошибочного решения относительно значения символа  $u_i$  при декодировании методом последовательного исключения. Отсюда следует, что классические полярные коды обеспечивают наименьшую возможную вероятность ошибки декодирования методом последовательного исключения среди полярных кодов.

Сложность алгоритма последовательного исключения составляет  $O(n \log_i(n))$  базовых операций, где базовая операция соответствует одному вычислению выражения (1.11).

### 1.3.1.2. Гауссовская аппроксимация

Вероятность ошибки декодирования методом последовательного исключения  $P^{(e)}$  (см. выражение (1.12)) определяется значениями вероятностей  $P_i$ ,  $i \notin \mathcal{F}$ . Вероятность  $P_i$  может быть вычислена по заданному распределению логарифмических отношений правдоподобия (ЛОПП) для символа  $u_i$

$$L_{0,i} = \log \frac{P(y_0^{n-1}, \hat{u}_0^{i-1} | u_i = 0)}{P(y_0^{n-1}, \hat{u}_0^{i-1} | u_i = 1)}. \quad (1.13)$$

В случае полярных кодов с ядром Арикана такие ЛОПП могут быть рекурсивно вычислены согласно [60]

$$L_{\lambda,i} = 2 \tanh^{-1} (\tanh(L_{\lambda+1,i}/2) \cdot \tanh(L_{\lambda+1,i+\eta}/2)), \quad (1.14)$$

$$L_{\lambda,i+\eta} = L_{\lambda+1,i} + (-1)^{\hat{v}_{\lambda,i}} L_{\lambda+1,i+\eta}, \quad (1.15)$$

где  $L_{m,i} = \log \frac{P(y_i|0)}{P(y_i|1)}$ ,  $0 \leq \lambda < m$ ,  $\eta = 2^{m-l-1}$ ,  $i \in \{2\eta h + j | 0 \leq h < 2^l, 0 \leq j < \eta\}$  и  $\hat{v}_{h,j}$  — оценка  $j$ -го промежуточного символа на  $h$ -ом слое:

$$\hat{v}_{h,j} = \begin{cases} \hat{u}_j, & h = 0, \\ \hat{v}_{h-1,j} \oplus \hat{v}_{h-1,j+2^{m-h}}, & ((j \bmod 2^{m-h-1}) < 2^{m-h}) \wedge (h > 0), \\ \hat{v}_{h-1,j}, & ((j \bmod 2^{m-h-1}) \geq 2^{m-h}) \wedge (h > 0). \end{cases}$$

Алгоритм последовательного исключения принимает решение  $\hat{u}_j = 1$  при  $L_{0,j} < 0$  и  $\hat{u}_j = 0$  в противном случае. Для анализа его корректирующей способности может использоваться обобщение предложенной в [21, 79] Гауссовской аппроксимации.

Рассмотрим случай передачи нулевого кодового слова, предполагая, что ЛОПП  $L_{\lambda,i}$ , задаваемые выражениями (1.14)–(1.15), являются Гауссовыми случайными величинами с математическими ожиданиями  $e_{\lambda,i} = E_{Y_0^{n-1}} [L_{\lambda,i}]$ , где  $Y_j$  — случайная величина, соответствующая  $j$ -му символу принятой последовательности. Из выражений, приведенных в [21], получаем, что эти значения должны

удовлетворять

$$e_{\lambda,i} = \phi^{-1} (1 - (1 - \phi(e_{\lambda+1,i}))(1 - \phi(e_{\lambda+1,i+\eta}))), \quad (1.16)$$

$$e_{\lambda,i+\eta} = e_{\lambda+1,i} + e_{\lambda+1,i+\eta}, \quad (1.17)$$

где  $i \in \{2\eta h + j | 0 \leq h < 2^\lambda, 0 \leq j < \eta\}$ ,  $\eta = 2^{m-\lambda-1}$ ,  $e_{m,j} = E_{y_j} \left[ \log \frac{W(y_j|0)}{W(y_j|1)} \right]$ , and

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

Вероятность ошибки  $p_{\lambda,i}$  в  $i$ -ом битовом подканале слоя  $\lambda$  может быть вычислена как  $p_{\lambda,i} = \Omega(e_{\lambda,i})$ , где  $\Omega(z) = Q\left(\sqrt{z/2}\right)$ . При описании метода, предложенного в разделе 2.2, удобно использовать выражения (1.16) и (1.17), переписанные в терминах вероятности ошибки, то есть

$$p_{\lambda,i+b\eta} = \Psi^{(b)}(p_{\lambda+1,i}, p_{\lambda+1,i+\eta}), \quad b \in \{0, 1\}, \quad (1.18)$$

где

$$\Psi^{(0)}(p_0, p_1) = \Omega \left( \phi^{-1} \left( 1 - \prod_{b=0}^1 (1 - \phi(\Omega^{-1}(p_b))) \right) \right), \quad (1.19)$$

$$\Psi^{(1)}(p_0, p_1) = \Omega \left( \sum_{b=0}^1 \Omega^{-1}(p_b) \right), \quad (1.20)$$

$0 \leq \lambda < m$ ,  $\eta = 2^{m-\lambda-1}$ ,  $i \in \{2\eta h + j | 0 \leq h < 2^\lambda, 0 \leq j < \eta\}$  и  $p_{m,j} = \Omega(4/N_0)$  в случае, если символы кодового слова передаются по каналу со спектральной плотностью мощности шума  $N_0$ . Полученные  $p_{0,i}$  используются в качестве  $P_i$  в выражении (1.12),  $0 \leq i < n$ .

### 1.3.2. Списочный алгоритм последовательного исключения

Вероятность ошибки декодирования при использовании алгоритма последовательного исключения значительно превосходит вероятность ошибки декодирования по максимуму правдоподобия. В [72] был предложен списочный алгоритм

последовательного исключения для полярных кодов с ядром Арикана, обладающий значительно большей корректирующей способностью, чем классический алгоритм последовательного исключения.

Основным недостатком алгоритма последовательного исключения является невозможность исправления ошибок, допущенных на ранних этапах работы алгоритма. Поскольку на  $i$ -ой фазе декодер учитывает только подмножество строк проверочной матрицы полярного кода, соответствующих замороженным символам с индексами  $i' : i' < i, i' \in \mathcal{F}$ , решения принимаемые декодером на ранних фазах оказываются ненадежными. При обработке новых проверок на четность может понадобиться корректировка этих решений. Одно из решений данной проблемы состоит в хранении списка наиболее вероятных путей заданной длины, где под путем длины  $i$  понимается последовательность  $u_0^{i-1}$  значений уже обработанных символов  $u_j$ . На каждой итерации алгоритма все пути в списке удлиняются на один элемент. На 0-ой итерации список состоит из одного пути нулевой длины. Если  $i \in \mathcal{F}$ , то при удлинении пути  $u_0^{i-1}$  на один элемент можно получить только  $u_0^i$  такой, что  $u_i = 0$ . В противном случае, для  $u_0^{i-1}$  необходимо рассмотреть 2 пути  $u_0^i$  для значений  $u_i \in \{0, 1\}$ . Пусть  $L$  — наибольшее количество путей, рассматриваемых на  $i$ -ой итерации. Если количество путей в списке превышает  $L$ , то в списке оставляют  $L$  путей с наибольшими вероятностями  $P(u_0^i | y_0^{n-1})$ , а прочие пути исключают из списка.

На Рис. 1.3 представлен график зависимости вероятности ошибки декодирования от отношения сигнал/шум для (2048, 1024) полярного кода и (2040, 1020) кода МППЧ, при декодировании которых использовались списочный алгоритм последовательного исключения и алгоритм распространения доверия, соответственно. Видно, что при увеличении размера списка  $L$  вероятность ошибки декодирования полярного кода снижается. Случай  $L = 1$  соответствует классическому алгоритму последовательного исключения, в то время как случаи  $L = 16$  или  $L = 32$  соответствуют декодированию полярных кодов почти по максимуму правдоподобия, однако, их корректирующая способность оказывается

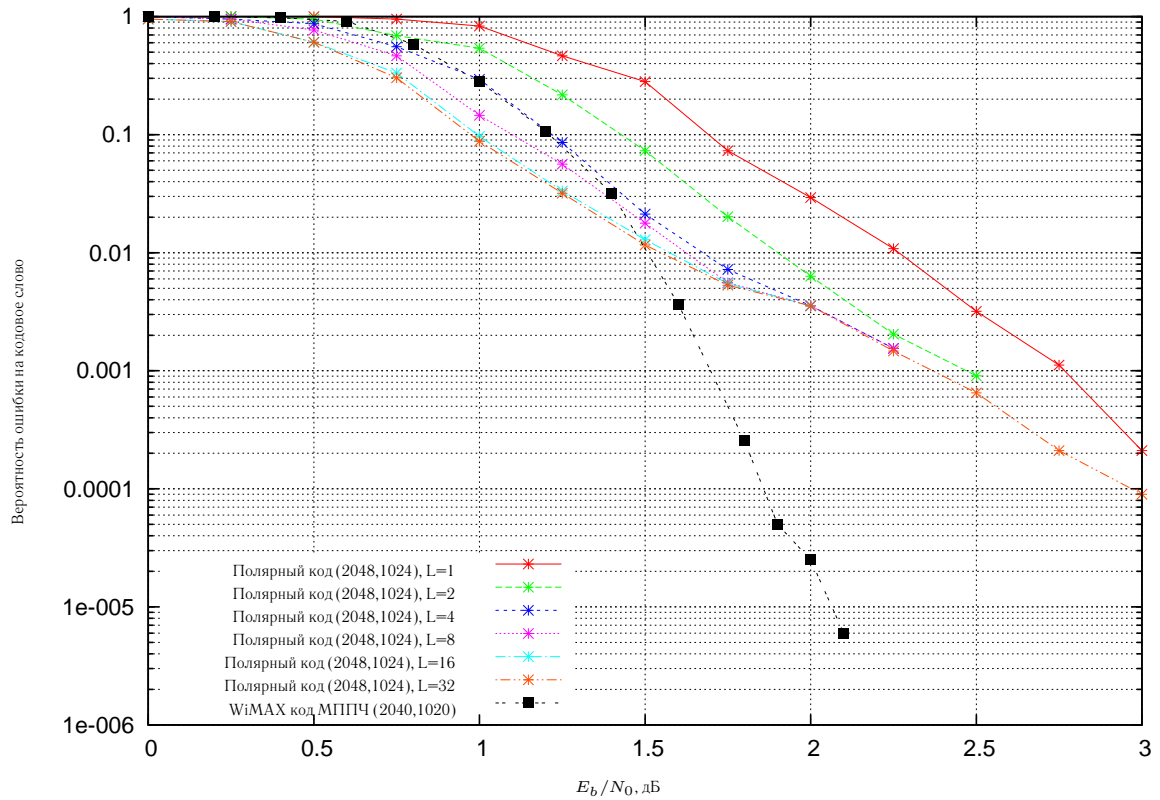


Рис. 1.3. Вероятность ошибки декодирования (2048, 1024) полярного кода

существенно ниже чем кодов МППЧ.

Вычислительная сложность списочного алгоритма последовательного исключения составляет  $O(Ln \log_l(n))$  базовых операций, где  $l$  – размерность ядра и  $n = l^m$  – длина кода.

### 1.3.3. Стековый алгоритм последовательного исключения

По сравнению со списочным алгоритмом последовательного исключения, стековый алгоритм, предложенный в [62], обладает меньшей сложностью при сохранении той же корректирующей способности.

Отличие стекового от списочного алгоритма состоит в том, что в стеке (приоритетной очереди) хранятся пути различных длин. На каждой итерации декодирования из стека выбирается путь  $u_0^{i-1}$  с наибольшей вероятностью  $P(u_0^{i-1}|y_0^{n-1})$ . Этот путь удлиняется на элемент  $u_i$ , для  $u_0^i$  вычисляется вероятность  $P(u_0^i|y_0^{n-1})$ , затем путь  $u_0^i$  помещается в стек вместе с вычисленной вероятностью. Как и в

случае списочного алгоритма, если  $i \in \mathcal{F}$ , то  $u_i = 0$ , иначе рассматриваются оба варианта значения  $u_i \in \{0, 1\}$ . Если число путей в стеке превосходит заданное ограничение  $\Theta$ , то пути с наименьшими вероятностями  $P(u_0^i | y_0^{n-1})$  исключаются из стека. Также используется ограничение на число путей фиксированной длины: если число путей длины  $i$  превосходит  $L$ , то наименее вероятные из них исключаются из стека, а также исключаются все пути, длина которых меньше  $i$ .

Для заданной принятой последовательности при  $\Theta \geq Ln$  и фиксированном  $L$  списочный и стековый алгоритмы в качестве результата декодирования вернут одно и то же кодовое слово. При этом вычислительная сложность стекового алгоритма окажется существенно меньше чем списочного алгоритма, что обусловлено тем, что на каждой итерации стекового алгоритма удлиняется только один наиболее вероятный путь, в то время как на каждой итерации списочного алгоритма удлиняются  $L$  наиболее вероятных путей. Заметим, что число итераций стекового алгоритма не превосходит число итераций списочного алгоритма, умноженное на  $L$ . Таким образом сложность стекового алгоритма последовательного исключения ограничена сверху сложностью списочного алгоритма последовательного исключения при фиксированном значении  $L$ .

Стековый алгоритм последовательного исключения можно рассматривать как обобщение стекового алгоритма, предложенного К.Ш. Зигангировым в [4] для случая произвольных древовидных кодов. К.Ш. Зигангиров рассматривал несколько вариаций стекового алгоритма. В первой их них декодирование символа  $u_i$  заканчивалось, если метрика оказывалась выше заданного порога, во второй — если длина пути достигала заданного значения. Также рассматривались схожие вариации с ограничением на емкость памяти декодера.

#### 1.3.4. Эффективная реализация

В данном разделе рассматривается эффективная реализация алгоритма последовательного исключения и его списочной версии, предложенная в [72], обоб-

```

SUCCESSIVECANCELLATION( $P(y_i|0), P(y_i|1), 0 \leq i < n$ )
1  for  $\beta \leftarrow 0$  to  $n - 1$ 
2  do  $P_0[\langle 0, \beta \rangle][v] \leftarrow P(y_\beta|v), \quad v \in \{0, 1\}$ 
3  for  $\phi \leftarrow 0$  to  $n - 1$ 
4  do RECURSIVECALCP( $m, \phi$ )
5    if ( $\phi \in \mathcal{F}$ )
6      then  $B_m[\langle \phi, 0 \rangle] \leftarrow 0$ 
7      else  $B_m[\langle \phi, 0 \rangle] \leftarrow \arg \max_{v \in \{0, 1\}} P_m[\langle \phi, 0 \rangle][v]$ 
8    if ( $\phi \bmod l = l - 1$ )
9      then RECURSIVEUPDATEB( $m, \phi$ )
10 return  $c_0^{n-1} \leftarrow (B_0[\langle 0, 0 \rangle], \dots, B_0[\langle 0, n - 1 \rangle])$ 

```

Рис. 1.4. Алгоритм последовательного исключения

щенная на случай полярных кодов с произвольным двоичным ядром поляризации.

### 1.3.4.1. Алгоритм последовательного исключения

В данном разделе приведено описание того, как для  $(n, k)$  полярного кода с  $l \times l$  ядром пространственная сложность алгоритма последовательного исключения  $O(n \log_l(n))$  может быть снижена до  $O(n)$ . При этом вычислительная сложность алгоритма последовательного исключения сохраняется.

Пусть  $\Lambda = 2^\lambda$  для слоя с номером  $\lambda, 0 \leq \lambda \leq m$ . Напомним, что вероятности  $P(u_0^i | y_0^{n-1})$ , необходимые для принятия решений относительно значений информационных символов, вычисляются согласно выражению (1.11). На  $\lambda$ -ом слое используется следующее представление индекса  $i = \phi + l^\lambda \beta$ , где  $0 \leq \phi < l^\lambda$  и  $0 \leq \beta < l^{m-\lambda}$ . Числа  $\phi$  и  $\beta$  обозначают фазу и номер ветви для индекса  $i$ , соответственно. Таким образом, на  $\lambda$ -ом слое есть  $l^{m-\lambda}$  ветвей, которые находятся на одной из  $l^\lambda$  фаз. Для краткости изложения используется обозначение  $\langle \phi, \beta \rangle_\lambda = \phi + l^\lambda \beta$ .

На Рис. 1.4-1.6 представлена непосредственная реализация метода после-

```

RECURSIVEUPDATEB( $\lambda, \phi$ )
1   $\psi \leftarrow \lfloor \phi/l \rfloor$ 
2  for  $\beta \leftarrow 0$  to  $l^{m-\lambda} - 1$ 
3  do  $(B_{\lambda-1}[\langle \psi, l\beta \rangle], \dots, B_{\lambda-1}[\langle \psi, l\beta + l - 1 \rangle]) \leftarrow (B_{\lambda}[\langle \phi - l + 1, \beta \rangle], \dots, B_{\lambda}[\langle \phi, \beta \rangle]) M$ 
4  if  $(\psi \bmod l = l - 1)$ 
5  then RECURSIVEUPDATEB( $\lambda - 1, \psi$ )

```

Рис. 1.5. Инициализация промежуточных символов

```

RECURSIVECALCP( $\lambda, \phi$ )
1  if  $(\lambda = 0)$ 
2  then return
3   $\psi \leftarrow \lfloor \phi/l \rfloor$ 
4  if  $(\phi \bmod l = 0)$ 
5  then RECURSIVECALCP( $\lambda - 1, \psi$ )
6   $t \leftarrow \phi \bmod l$ 
7  for  $\beta \leftarrow 0$  to  $l^{m-\lambda} - 1$ 
8  do  $u_0^{t-1} \leftarrow (B_{\lambda}[\langle l\psi, \beta \rangle], \dots, B_{\lambda}[\langle \phi - 1, \beta \rangle])$ 
9   $P_{\lambda}[\langle \phi, \beta \rangle] \leftarrow \sum_{u_t^{l-1} \in \{0,1\}^{l-t}} \prod_{j=0}^{l-1} P_{\lambda-1}[\langle \psi, l\beta + j \rangle] [(u_0^{l-1} M)_j]$ 

```

Рис. 1.6. Вычисление вероятностей для промежуточных символов

довательного исключения. В ней используются массивы вероятностей  $P_{\lambda}$  и массивы значений символов  $B_{\lambda}$  длины  $l^m$ ,  $0 \leq \lambda \leq m$ . Каждый элемент  $P_{\lambda}[i]$  состоит из пары вероятностей  $P_{\lambda}[i][0]$  и  $P_{\lambda}[i][1]$ , в то время как  $B_{\lambda}[i] \in \{0, 1\}$ ,  $0 \leq i < l^m$ . Заметим, что инициализация элементов этих массивов выполняется в процессе декодирования. Пространственная сложность алгоритма составляет  $O(n \log_l(n))$ . Для краткости обозначение  $T_{\lambda}[\langle \phi, \beta \rangle_{\lambda}]$  заменяется на  $T_{\lambda}[\langle \phi, \beta \rangle]$  для некоторого массива  $T$ .

Рассмотрим преобразование описанной выше реализации алгоритма последовательного исключения, позволяющее снизить количество потребляемой памяти с  $O(n \log_l(n))$  до  $O(n)$ . Из Рис. 1.4 видно, что на фазе  $\phi$  вероятности



EFFICIENTSUCCESSIVEANCELLATION( $P(y_i|0), P(y_i|1), 0 \leq i < n$ )

```

1  for  $\beta \leftarrow 0$  to  $n - 1$ 
2  do  $P_0[\beta][v] \leftarrow P(y_\beta|v), \quad v \in \{0, 1\}$ 
3  for  $\phi \leftarrow 0$  to  $n - 1$ 
4  do EFFICIENTRECURSIVECALCP( $m, \phi$ )
5    if ( $\phi \in \mathcal{F}$ )
6      then  $C_m[0][\phi \bmod l] \leftarrow 0$ 
7      else  $C_m[0][\phi \bmod l] \leftarrow \arg \max_{v \in \{0,1\}} P_m[0][v]$ 
8    if ( $\phi \bmod l = l - 1$ )
9      then RECURSIVEUPDATEC( $m, \phi$ )
10 return  $c_0^{n-1} \leftarrow (C_0[0][0], \dots, C_0[n-1][0])$ 

```

Рис. 1.7. Алгоритм последовательного исключения при малом потреблении памяти

$P_m[\langle \phi', 0 \rangle]$  для  $\phi' < \phi$  не будут переиспользованы. С другой стороны, вероятности  $P_m[\langle \phi'', 0 \rangle]$  для  $\phi'' > \phi$  не проинициализированы. Таким образом, на фазе  $\phi$  на  $m$ -ом слое работа ведется только с вероятностью  $P_m[\langle \phi, 0 \rangle]$ , исходя из чего, можно пренебречь фазой  $\phi$  и использовать только элемент  $P_m[\langle 0 \rangle]$ , исключая из рассмотрения все прочие элементы массива  $P_m$ . Такой подход может быть реализован и для других слоев  $\lambda$ , то есть для  $0 \leq \lambda \leq m$  понадобится только  $l^{m-\lambda}$  элементов массива  $P_\lambda$ . Соответственно, обозначение  $P_\lambda[\langle \phi, \beta \rangle]$  можно заменить на  $P_\lambda[\beta]$ .

Схожие действия могут быть выполнены над массивом  $B$ , однако, в случае этого массива нельзя исключить из рассмотрения фазу. Для  $0 \leq \lambda \leq m$  определим массив  $C_\lambda$  длины  $n/l$ , состоящий из векторов длины  $l$ . Заменяем  $B_\lambda[\langle \phi, \beta \rangle]$  на  $C_\lambda[\psi + \beta l^{\lambda-1}][\phi \bmod l]$ , где  $\psi = \lfloor \phi/l \rfloor$ . Теперь индекс  $\psi$  может быть отброшен, в результате чего, мы переходим от обозначения  $C_\lambda[\psi + \beta l^{\lambda-1}][\phi \bmod l]$  к  $C_\lambda[\beta][\phi \bmod l]$ . Таким образом, массив  $C_\lambda$  состоит из  $l^{m-\lambda}$  векторов длины  $l$ .

Рис. 1.7-1.8 иллюстрируют получаемую реализацию алгоритма последовательного исключения. Функция *EfficientRecursiveCalcP* отличается от *RecursiveCalcP*, представленной на Рис. 1.6, только заменой стро-

RECURSIVEUPDATEC( $\lambda, \phi$ )

```

1   $\psi \leftarrow \lfloor \phi/l \rfloor$ 
2  for  $\beta \leftarrow 0$  to  $l^{m-\lambda} - 1$ 
3  do  $(C_{\lambda-1}[\beta][\psi \bmod l], \dots, C_{\lambda-1}[\beta + l - 1][\psi \bmod l]) \leftarrow (C_{\lambda}[\beta][0], \dots, C_{\lambda}[\beta][l - 1]) M$ 
4  if  $(\psi \bmod l = l - 1)$ 
5  then RECURSIVEUPDATEC( $\lambda - 1, \psi$ )

```

Рис. 1.8. Инициализация промежуточных символов при малом потреблении памяти

ки 8 на  $u_0^{t-1} \leftarrow (C_{\lambda}[\beta][0], \dots, C_{\lambda}[\beta][t - 1])$  и строки 9 на  $P_{\lambda}[\beta] \leftarrow \sum_{u_t^{l-1} \in \{0,1\}^{l-t}} \prod_{j=0}^{l-1} P_{\lambda-1}[l\beta + j] \left[ (u_0^{l-1} M)_j \right]$ .

#### 1.3.4.2. Списочный алгоритм последовательного исключения

Списочный алгоритм последовательного исключения предполагает итеративное построение  $L$  наиболее вероятных путей, при этом на каждой итерации длина каждого пути увеличивается на один. При непосредственной реализации алгоритма выполняется копирование структур данных при каждом разветвлении пути. Поскольку число разветвлений составляет  $O(Ln)$  и размер структур данных как минимум  $O(n)$ , то операция копирования занимает  $O(Ln^2)$ . В [72] показано, как сложность может быть снижена до  $O(Ln \log_l(n))$ .

Сложность копирования массивов  $P_{\lambda}$ , состоящих из  $l^{m-\lambda}$  элементов, растет экспоненциально с уменьшением  $\lambda$ . Из Рис. 1.7 видно, что обращение к массиву  $P_{\lambda}$  осуществляется только один раз при увеличении  $\phi$  на  $2^{m-\lambda}$ . Для уменьшения числа копирований используется техника “ленивого копирования”, согласно которой один массив может соответствовать более чем одному пути. При этом, реальное копирование данных осуществляется непосредственно перед следующим удлинением этого пути. Аналогичный подход применим и для массива  $C$ .

На Рис. 1.9, 1.10, 1.11 и 1.13 представлена реализация списочного алгоритма последовательного исключения, предложенная в [72]. В ал-

ASSIGNINITIALPATH()

```
1  InactivePathIndices ← new queue
2  InactiveArrayIndices ← new queue [ $m + 1$ ]
3  ActivePath ← new boolean [ $L$ ]
4  ArrayReferenceCount ← new integer [ $m + 1$ ][ $L$ ]
5  PathIndexToArrayIndex ← new integer [ $m + 1$ ][ $L$ ]
6  ArrayPointerP ← new pointer [ $m + 1$ ][ $L$ ]
7  ArrayPointerC ← new pointer [ $m + 1$ ][ $L$ ]
8  for  $\lambda \leftarrow 0$  to  $m$ 
9  do for  $s \leftarrow 0$  to  $L - 1$ 
10     do ArrayPointerP $[\lambda][s]$  ← new float [ $q$ ]
11         ArrayPointerC $[\lambda][s]$  ← new boolean [ $l^{m-\lambda}$ ][ $L$ ]
12         ArrayReferenceCount $[\lambda][s]$  ← 0
13         PUSH(InactiveArrayIndices $[\lambda], s$ )
14 for  $f \leftarrow 0$  to  $L - 1$ 
15 do ActivePath $[f]$  ← false
16     PUSH(InactivePathIndices,  $f$ )
17  $f \leftarrow$  POP(InactivePathIndices)
18 ActivePath $[f]$  ← true
19 for  $\lambda \leftarrow 0$  to  $m$ 
20 do  $s \leftarrow$  POP(InactiveArrayIndices $[\lambda]$ )
21     PathIndexToArrayIndex $[\lambda][f]$  ←  $s$ 
22     ArrayReferenceCount $[\lambda][s]$  ← 1
23 return  $f$ 
```

Рис. 1.9. Инициализация списочного декодера последовательного исключения

```

CONTINUEPATHSFROZENSYMBOL( $\phi$ )
1  for  $f \leftarrow 0$  to  $L - 1$ 
2  do if  $PathIndexInactive(f)$ 
3      then continue
4       $C_m \leftarrow GETARRAYPOINTERC(m, f)$ 
5       $C_m[0][\phi \bmod l] \leftarrow 0$ 

```

Рис. 1.10. Удлинение путей на один замороженный символ

горитме используется процедура  $RecursiveUpdateC(f, \lambda, \phi)$ , совпадающая с  $RecursiveUpdateC(\lambda, \phi)$ , приведенной на Рис. 1.8, за исключением того, что она выполняется для заданного  $f$ , то есть используются массивы:  $C_{\lambda-1} = GetArrayPointerC(\lambda - 1, f)$  и  $C_\lambda = GetArrayPointerC(\lambda, f)$ . На Рис. 1.9 функции  $Push$  и  $Pop$  предназначены для добавления в стек элемента и извлечения из нее элемента, соответственно. Функция  $Rearrange(ForksArray, \rho)$  на Рис. 1.11 выполняет такую перестановку элементов массива  $ForksArray$ , что для любых  $\alpha < \rho$  и  $\beta \geq \rho$  справедливо  $ForksArray[\alpha][0] \geq ForksArray[\beta][0]$ . Целью этой перестановки является выявление  $\rho$  путей с наибольшими вероятностями, построение которых продолжится на последующих шагах алгоритма. Прочие пути исключаются из рассмотрения посредством вызова функции  $KillPath(f)$ , освобождающей структуры данных, соответствующие  $f$ -му пути.

## 1.4. Коды Рида-Соломона

Коды Рида-Соломона являются недвоичными линейными кодами с максимально достижимым минимальным расстоянием. Кодовыми словами  $(n, k, n - k + 1)$  кода Рида-Соломона являются вектора вида  $(f(x_0), \dots, f(x_{n-1}))$ , элементами которых являются значения полинома  $f(x) = \sum_{i=0}^{k-1} f_i x^i$ , вычисленные в различных точках  $x_i \in \mathbb{F}_q$ . Коэффициентами полинома  $f(x)$  являются символы информационной последовательности  $(f_0, \dots, f_{k-1}) \in \mathbb{F}_q^k$ .

```

CONTINUEPATHSUNFROZENSYMBOL( $\phi$ )
1  ForksArray  $\leftarrow$  new  $\langle$  float, symbol, index  $\rangle$  [lL]
2  i  $\leftarrow$  0
3  for f  $\leftarrow$  0 to L - 1
4  do if (PathIndexInactive(f) = false)
5      then  $P_m \leftarrow$  GETARRAYPOINTERP(m, f)
6          ForksArray[li + t]  $\leftarrow$  ( $P_m[0][t]$ , t, f),  $0 \leq t < l$ 
7          i  $\leftarrow$  i + 1
8   $\rho \leftarrow$  min(li, L)
9  ForksArray  $\leftarrow$  REARRANGE(ForksArray,  $\rho$ )
10 ContForks  $\leftarrow$  new boolean [L][q]
11 for r  $\leftarrow$  0 to  $\rho - 1$ 
12 do f  $\leftarrow$  ForksArray[r][2]
13     v  $\leftarrow$  ForksArray[r][1]
14     ContForks[f][v]  $\leftarrow$  true
15 for f  $\leftarrow$  0 to L - 1
16 do if (PathIndexInactive(f) = false)
17     then if (ContForks[f][0] = false) and (ContForks[f][1] = false)
18         then KILLPATH(f)
19 for f  $\leftarrow$  0 to L - 1
20 do if (ContForks[f][0] = true) or (ContForks[f][1] = true)
21     then  $C_m \leftarrow$  GETARRAYPOINTERC(m, f)
22         if (ContForks[f][0] = true) and (ContForks[f][1] = true)
23             then  $C_m[0][\phi \bmod l] \leftarrow$  0
24                  $f' \leftarrow$  CLONEPATH(f)
25                  $C_m \leftarrow$  GETARRAYPOINTERC(m, f')
26                  $C_m[0][\phi \bmod l] \leftarrow$  1
27         else if (ContForks[f][0] = true)
28             then  $C_m[0][\phi \bmod l] \leftarrow$  0
29         else  $C_m[0][\phi \bmod l] \leftarrow$  1

```

Рис. 1.11. Удлинение путей на один незамороженный символ

```

LISTEFFICIENTRECURSIVECALCP( $\lambda, \phi, f$ )
1  if ( $\lambda = 0$ )
2    then return
3   $\psi \leftarrow \lfloor \phi/l \rfloor$ 
4  if ( $\phi \bmod l = 0$ )
5    then EFFICIENTRECURSIVECALCP( $\lambda - 1, \psi, f$ )
6   $P_\lambda \leftarrow \text{GETARRAYPOINTERP}(\lambda, f)$ 
7   $P_{\lambda-1} \leftarrow \text{GETARRAYPOINTERP}(\lambda - 1, f)$ 
8   $t \leftarrow \phi \bmod l$ 
9  for  $\beta \leftarrow 0$  to  $l^{m-\lambda} - 1$ 
10 do  $u_0^{t-1} \leftarrow (C_\lambda[\beta][0], \dots, C_\lambda[\beta][t-1])$ 
11    $P_\lambda[\beta] \leftarrow \sum_{u_i^{l-1} \in \{0,1\}^{l-t}} \prod_{j=0}^{l-1} P_{\lambda-1}[l\beta + j] \left[ (u_0^{l-1}M)_j \right]$ 

```

Рис. 1.12. Вычисление вероятностей для промежуточных символов заданного пути

## 1.5. Декодирование кодов Рида-Соломона

Одним из первых методов мягкого декодирования является декодирование по обобщенному минимальному расстоянию [31], однако, этот метод не имел значительных преимуществ перед жестким декодированием. Намного больший энергетический выигрыш может быть достигнут при использовании алгоритма Кёттера-Варди [42].

### 1.5.1. Алгоритм Кёттера-Варди

Алгоритм Кёттера-Варди, предложенный в [42], состоит из трех основных шагов:

1. Построение  $n \times q$  матрицы кратностей корней  $M$  для заданной принятой последовательности. Элементами  $M_{i,j}$  являются неотрицательные целые числа, характеризующие вероятность того, что  $i$ -ый символ кодового слова равен  $j$ -му элементу поля  $\mathbb{F}_q$ . При построении матрицы кратностей кор-

LISTEFFICIENTSUCCESSIVECANCELLATION( $L, P(y_i|v), 0 \leq i < n, v \in \mathbb{F}_2$ )

```

1   $f \leftarrow \text{ASSIGNINITIALPATH}()$ 
2   $P_0 \leftarrow \text{GETARRAYPOINTERP}(0, f)$ 
3  for  $\beta \leftarrow 0$  to  $n - 1$ 
4  do  $P_0[\beta][v] \leftarrow P(y_\beta|v), \quad v \in \mathbb{F}_2$ 
5  for  $\phi \leftarrow 0$  to  $n - 1$ 
6  do for  $f \leftarrow 0$  to  $L - 1$ 
7      do if ( $\text{PATHINDEXINACTIVE}(f)$ )
8          then continue
9          LISTEFFICIENTRECURSIVECALCP( $m, \phi, f$ )
10         if ( $\phi \in \mathcal{F}$ )
11             then CONTINUEPATHSFROZENSYMBOL( $\phi$ )
12             else CONTINUEPATHSUNFROZENSYMBOL( $\phi$ )
13         if ( $\phi \bmod l = l - 1$ )
14             then for  $f \leftarrow 0$  to  $L - 1$ 
15                 do if ( $\text{PATHINDEXINACTIVE}(f)$ )
16                     then continue
17                 RECURSIVEUPDATEC( $f, m, \phi$ )
18   $f' \leftarrow 0$ 
19   $p' \leftarrow 0$ 
20  for  $f \leftarrow 0$  to  $L - 1$ 
21  do if ( $\text{PATHINDEXINACTIVE}(f)$ )
22      then continue
23       $C_m \leftarrow \text{GETARRAYPOINTERC}(m, f)$ 
24       $P_m \leftarrow \text{GETARRAYPOINTERP}(m, f)$ 
25      if ( $p' < P_m[0][C_m[0][1]]$ )
26          then  $f' \leftarrow f$ 
27               $p' \leftarrow P_m[0][C_m[0][1]]$ 
28   $C_0 \leftarrow \text{GETARRAYPOINTERC}(0, f')$ 
29  return  $c_0^{n-1} \leftarrow (C_0[0][0], \dots, C_0[n-1][0])$ 

```

Рис. 1.13. Списочный алгоритм последовательного исключения

ней может быть использован как метод, описанный в [42], так и один из методов, предложенных в [22, 27, 38, 41, 65].

2. Интерполяция по множеству точек  $(x_i, y_j)$ ,  $0 \leq i < n$ ,  $0 \leq j < q$ . При этом интерполяционный полином должен иметь минимально возможную  $(1, k - 1)$ -взвешенную степень и корни кратности как минимум  $M_{i,j}$  во всех точках  $(x_i, y_j)$ .  $(a, b)$ -взвешенная степень одночлена  $cx^i y^j$  равна  $ai + bj$ . Под  $(a, b)$ -взвешенной степенью полинома  $\text{wdeg}_{(a,b)} Q(x, y)$  понимается  $(a, b)$ -взвешенная степень старшего члена этого полинома, где старшим членом полинома  $Q(x, y)$  является его одночлен  $cx^i y^j$  с наибольшей  $(a, b)$ -взвешенной степенью. Точка  $(x_i, y_j)$  является корнем полинома  $Q(x, y)$  кратности  $M_{i,j}$ , если соответствующие производные Хассе  $Q(x, y)$  удовлетворяют условию

$$Q^{[\alpha, \beta]}(x_i, y_j) = 0, \quad \alpha + \beta < M_{i,j}. \quad (1.21)$$

Производная Хассе определена как

$$Q^{[\alpha, \beta]}(x_i, y_j) = \sum_{\alpha' \geq \alpha} \sum_{\beta' \geq \beta} \binom{\alpha'}{\alpha} \binom{\beta'}{\beta} q_{\alpha', \beta'} x_i^{\alpha' - \alpha} y_j^{\beta' - \beta}.$$

В дальнейшем, для корней  $(x_i, y_j)$  кратности  $M_{i,j}$  будет использоваться обозначение  $Q(x_i, y_j) = 0^{M_{i,j}}$ .

3. Факторизация полинома  $Q(x, y)$ . Среди полиномов с коэффициентами из  $\mathbb{F}_q$  и степенью, не превосходящей  $(k - 1)$ , осуществляется поиск полиномов  $\omega(x)$ , являющихся корнями полинома  $Q(x, y)$  (то есть  $y - \omega(x)$  делит  $Q(x, y)$ ). Для каждого найденного  $\omega(x)$  строится соответствующее кодовое слово. Из полученного списка кодовых слов выбирается наиболее вероятное, являющееся результатом работы алгоритма. Факторизация полинома  $Q(x, y)$  может быть эффективно реализована с помощью рекурсивного алгоритма [67].



Частным случаем алгоритма Кёттера-Варди является алгоритм Гурусвами-Судана, предложенный в [33]. В данном алгоритме рассматривается только  $n$  интерполяционных точек  $(x_i, y_j)$ , имеющих различные значения  $x_i$ , при этом значения  $M_{i,j}$  для этих точек совпадают. Алгоритм Гурусвами-Судана позволяет найти все кодовые слова, совпадающие с вектором жестких решений  $(h_0, \dots, h_{n-1})$  не менее чем в  $\sqrt{n(k-1)}$  позициях, где  $h_i = \arg \max_{v \in \mathbb{F}_q} P(y_i|v)$ .

Следует отметить, что метод Кёттера-Варди не обеспечивает декодирование по максимуму правдоподобия, поэтому актуальной является задача построения для кодов Рида-Соломона алгоритмов декодирования с большей корректирующей способностью.

### 1.5.2. Двумерная интерполяция

При построении списка наиболее вероятных кодовых слов с помощью алгоритма Кёттера-Варди наиболее трудоемким этапом является построение многочлена от двух переменных наименьшей степени, имеющего корни  $(x_i, y_j)$  кратности  $M_{i,j}$ ,  $0 \leq i < n$ ,  $0 \leq j < q$ . Данный многочлен может быть найден в базисе Грёбнера идеала многочленов, имеющих точки  $(x_i, y_j)$  корнями кратности  $M_{i,j}$ .

В рассматриваемых интерполяционных алгоритмах используются идеалы над кольцом  $\mathbb{F}_q[x, y]$ , являющимся кольцом полиномов от переменных  $x$  и  $y$ , коэффициенты которых принадлежат полю  $\mathbb{F}_q$ . Идеалом, порожденным полиномами  $Q_i(x, y) \in \mathbb{F}_q[x, y]$ ,  $0 \leq i \leq v$ , называется множество  $\langle Q_0(x, y), \dots, Q_v(x, y) \rangle = \left\{ \sum_{i=0}^v \omega_i(x, y) Q_i(x, y) \mid \omega_i(x, y) \in \mathbb{F}_q[x, y] \right\}$ . Модулем, порожденным полиномами  $Q_i(x, y) \in \mathbb{F}_q[x, y]$ ,  $0 \leq i \leq v$ , называется множество  $[Q_0(x, y), \dots, Q_v(x, y)] = \left\{ \sum_{i=0}^v \omega_i(x) Q_i(x, y) \mid \omega_i(x) \in \mathbb{F}_q[x] \right\}$ . Множество  $\{Q_0(x, y), \dots, Q_v(x, y)\} \subset I$  называется базисом Грёбнера идеала  $I$  тогда и только тогда, когда старший член любого полинома из  $I$  делится на старший член хотя бы одного  $Q_i(x, y)$ . Аналогичное определение базиса Грёбнера справедливо и для модуля.

### 1.5.2.1. Итеративный интерполяционный алгоритм

Итеративный интерполяционный алгоритм, предложенный в [61], предназначен для построения базиса Грёбнера модуля  $I_{M,\rho} = \{Q(x, y) \in \mathbb{F}_q[x, y] \mid Q(x_i, y_j) = 0^{M_{i,j}}, 0 \leq i < n, 0 \leq j < q, \text{wdeg}_{(0,1)} Q(x, y) < \rho\}$ . Такой базис Грёбнера содержит искомый интерполяционный полином наименьшей  $(1, k - 1)$ -взвешенной степени при условии того, что  $\rho$  имеет достаточно большое значение.

Рис. 1.14 иллюстрирует работу итеративного интерполяционного алгоритма. При инициализации базиса  $(B_0, \dots, B_\rho)$  учитывается, что все степени по  $y$  старших членов полиномов должны быть различными, при этом выбираются наименьшие возможные степени по  $x$ , то есть нулевые (строки 1 — 2). Данное ограничение на степени старших членов полиномов базиса Грёбнера сохраняется при дальнейшей обработке полиномов. Обработка заключается в том, что на каждой итерации алгоритма (строки 7 — 12) обеспечивается равенство нулю соответствующей производной Хассе  $B_h^{[\alpha,\beta]}(x, y)$  в заданной интерполяционной точке  $(x_i, y_j)$ . Для этого из полиномов базиса вычитается полином  $B_{h_0}(x, y)$ , домноженный на необходимый коэффициент. В качестве  $B_{h_0}(x, y)$  выбирается полином с наименьшей  $(1, k - 1)$ -взвешенной степенью, для которого  $B_{h_0}^{[\alpha,\beta]}(x_i, y_j) \neq 0$ . Такой выбор полинома  $B_{h_0}$  позволяет сохранить старшие члены остальных полиномов базиса. После вычитания полином  $B_{h_0}$  домножается на  $(x - x_i)$ . Базис Грёбнера искомого модуля получен, если для всех точек  $(x_i, y_j), 0 \leq i < n, 0 \leq j < q$ , выполняется  $B_h^{[\alpha,\beta]}(x_i, y_j) = 0, 0 \leq h < \rho, \alpha + \beta < M_{i,j}$ . Из полученного базиса выбирается полином с наименьшей  $(1, k - 1)$ -взвешенной степенью.

Число  $\rho$  вычисляется на основании параметров кода и матрицы кратностей корней, оно должно быть таким, что искомый интерполяционный полином гарантированно принадлежит соответствующему модулю. Сложность данного алгоритма может быть оценена как  $O(n^2 m^4 \rho)$ , где величина  $m$  равна максимальному

ITERATIVEINTERPOLATE( $M$ )

```

1  for  $i \leftarrow 0$  to  $\rho - 1$ 
2  do  $B_i(x, y) \leftarrow y^i$ 
3  for  $i \leftarrow 0$  to  $n - 1$ 
4  do for  $j \leftarrow 0$  to  $q - 1$ 
5    do for  $\alpha \leftarrow 0$  to  $M_{i,j} - 1$ 
6      do for  $\beta \leftarrow 0$  to  $M_{i,j} - \alpha - 1$ 
7        do for  $h \leftarrow 0$  to  $\rho - 1$ 
8          do  $\Delta_h \leftarrow B_h^{[\alpha, \beta]}(x_i, y_j)$ 
9           $h_0 \leftarrow \arg \min_{h: \Delta_h \neq 0} B_h(x, y)$ 
10         for  $h \neq h_0$ 
11           do  $B_h(x, y) \leftarrow B_h(x, y) - \frac{\Delta_h}{\Delta_{h_0}} B_{h_0}(x, y)$ 
12            $B_{h_0}(x, y) \leftarrow B_{h_0}(x, y)(x - x_i)$ 
13  return  $\min_{0 \leq h \leq \rho - 1} B_h(x, y)$ 

```

Рис. 1.14. Итеративный интерполяционный алгоритм

элементу матрицы кратностей корней.

### 1.5.2.2. Двоичный интерполяционный алгоритм

Двоичный интерполяционный алгоритм был предложен в [75] для реализации интерполяционного шага алгоритма Гурусвами-Судана. Таким образом, данный алгоритм предполагает, что  $n$  элементов  $M_{i,j}$  с различными  $i$  равны некоторому  $r$ , а прочие  $M_{i,j} = 0$ . Для простоты изложения, для точек  $(x_i, y_j)$  с  $M_{i,j} = r$  будем использовать обозначение  $(x_i, y_i)$ . В основе двоичного интерполяционного алгоритма лежит применение двоичного метода возведения в степень к идеалам.

Идеал  $I_r = \{Q(x, y) \in \mathbb{F}_q[x, y] \mid Q(x_i, y_i) = 0^r, 0 \leq i < n\}$  можно представить как

$$I_r = I_1^r = (\dots ((I_1^2 \cdot I_1^{r_{m-1}})^2 \cdot I_1^{r_{m-2}})^2 \cdot I_1^{r_{m-3}} \dots I_1^{r_1})^2 \cdot I_1^{r_0},$$

где  $r = \sum_{j=0}^m r_j 2^j$ ,  $r_m = 1$ ,  $I^2 = I \cdot I$ ,  $I^0 = \mathbb{F}_q[x, y]$ , и  $I \cdot \mathbb{F}_q[x, y] = I$ . Данное равенство справедливо, поскольку при перемножении идеалов кратности корней

многочленов складываются, то есть  $I_{r_1+r_2} = I_{r_1} \cdot I_{r_2}$ .

Рассмотрим способы построения произведения идеалов  $I' = \langle T_0(x, y), \dots, T_u(x, y) \rangle$  и  $I'' = \langle S_0(x, y), \dots, S_v(x, y) \rangle$ . Стандартный метод вычисления базиса идеала  $I' \cdot I'' = \langle T_i(x, y)S_j(x, y), 0 \leq i \leq u, 0 \leq j \leq v \rangle$  предполагает вычисление всех попарных произведений полиномов из базисов перемножаемых идеалов. Заметим, что многие из этих попарных произведений являются избыточными.

Для снижения числа умножений полиномов перейдем от задачи построения базиса Грёбнера идеала к задаче построения базиса Грёбнера модуля. Определим функцию

$$\Delta(\mathcal{B}) = \sum_{j=0}^s t_j,$$

где  $\mathcal{B} = (B_0(x, y), \dots, B_s(x, y))$  — базис Грёбнера некоторого модуля такой, что  $\text{LT}(B_j(x, y)) = a_j x^{t_j} y^j$ , где  $\text{LT}(B_j(x, y))$  — старший член полинома  $B_j(x, y)$ . В [75] показано, что если  $\mathcal{B} = (B_0(x, y), \dots, B_{\rho-1}(x, y))$  является базисом Грёбнера модуля  $I_{r,\rho} = \left\{ Q(x, y) = \sum_{j=0}^{\rho-1} q_j(x) y^j \mid Q(x, y) \in I_r \right\}$ , то

$$\Delta(\mathcal{B}) = \frac{nr(r+1)}{2}. \quad (1.22)$$

Необходимым условием того, что полиномы  $\mathcal{B} = (B_0(x, y), \dots, B_{\rho-1}(x, y))$  составляют базис Грёбнера модуля  $I_{r,\rho} = \left\{ \sum_{j=0}^{\rho-1} B_j(x, y) a_j(x) \mid a_j(x) \in \mathbb{F}_q[x] \right\}$ , является то, что  $\text{udeg } B_j(x, y) = \text{wdeg}_{(0,1)} \text{LT}(B_j(x, y))$  принимает различные значения при  $0 \leq j < \rho$ .

Если полиномы  $T_j(x, y)$ ,  $0 \leq j \leq u$ , удовлетворяют условиям:  $T_j(x_i, y_i) = 0^r$  для  $0 \leq i < n$ ,  $\text{LT } T_j(x, y) = a_j x^{t_j} y^j$ ,  $\text{wdeg}_{(0,1)} T_j(x, y) \leq u$ ,  $t_u = 0$  и

$$\Delta((T_0(x, y), \dots, T_u(x, y))) = \frac{nr(r+1)}{2},$$

то полиномы  $(T_0(x, y), \dots, T_u(x, y))$  составляют базис Грёбнера идеала  $I_r$ .

Заметим, что может существовать базис Грёбнера идеала  $I_r$ , который не удовлетворяет данным условиям. Пусть  $(T_0(x, y), \dots, T_u(x, y))$  и

```

REDUCE(( $B_0(x, y), \dots, B_{i-1}(x, y)$ ),  $Q(x, y)$ )
1   $B_i(x, y) \leftarrow Q(x, y)$ 
2  while  $\exists j : (0 \leq j < i) \wedge (\text{ydeg } B_j(x, y) = \text{ydeg } B_i(x, y))$ 
3  do if  $\text{LT}(B_i(x, y)) | \text{LT}(B_j(x, y))$ 
4      then  $W(x, y) \leftarrow B_j(x, y) - \frac{\text{LT}(B_j(x, y))}{\text{LT}(B_i(x, y))} B_i(x, y)$ 
5           $\text{LT}(B_j(x, y)) \leftarrow \text{LT}(B_i(x, y))$ 
6           $\text{LT}(B_i(x, y)) \leftarrow W(x, y)$ 
7      else  $\text{LT}(B_i(x, y)) \leftarrow \text{LT}(B_i(x, y)) - \frac{\text{LT}(B_i(x, y))}{\text{LT}(B_j(x, y))} B_j(x, y)$ 
8  if  $B_i(x, y) = 0$ 
9      then  $i \leftarrow i - 1$ 
10 return ( $B_0(x, y), \dots, B_i(x, y)$ )

```

Рис. 1.15. Многомерный алгоритм Евклида

$(S_0(x, y), \dots, S_v(x, y))$  — базисы Грёбнера идеалов  $I_{r_1}$  и  $I_{r_2}$ , соответственно, которые удовлетворяют вышеупомянутым условиям. Опишем процедуру построения произведения данных идеалов, используемую в двоичном интерполяционном алгоритме. Рассмотрим последовательность различных пар целых чисел  $(u'_j, v'_j)$ ,  $0 \leq j < (u + 1)(v + 1)$ , таких, что  $0 \leq u'_j \leq u, 0 \leq v'_j \leq v$  и  $\text{LT}(T_{u'_j}(x, y)S_{v'_j}(x, y)) = a_j x^{t_j} y^j$  для  $0 \leq j \leq u + v$ . Пусть

$$\mathcal{B}^{(u+v)} = (T_{u'_j}(x, y)S_{v'_j}(x, y), j = 0 \dots u + v) \quad (1.23)$$

является базисом некоторого подмодуля  $\mathcal{N}^{(0)}$  модуля  $I_{r_1+r_2, u+v+1}$ . Заметим, что данный базис подмодуля  $\mathcal{N}^{(0)}$  является базисом Грёбнера, и можно показать, что  $\Delta(\mathcal{B}^{(u+v)}) = \sum_{j=0}^{u+v} t_j \geq \frac{n(r_1 + r_2)(r_1 + r_2 + 1)}{2}$ .

В случае  $j > u + v$  модуль  $\mathcal{B}^{(j-1)}$  можно дополнить полиномом  $(T_{u'_j}(x, y)S_{v'_j}(x, y))$  с помощью алгоритма *Reduce*, приведенного на Рис. 1.15. Если полиномы  $B_i^{(j-1)}(x, y)$ ,  $0 \leq i < j$ , удовлетворяют условиям  $\text{LT } B_i^{(j-1)}(x, y) = a_i x^{t_i} y^i$  и  $\text{wdeg}_{(0,1)} B_i^{(j-1)}(x, y) < j$ , то алгоритм *Reduce* построит последовательность базисов  $\mathcal{B}^{(j)}$ ,  $j = 0, 1, \dots$ , модулей

$$\mathcal{N}^{(j)} = \left\{ Q(x, y) + a(x)T_{u'_j}(x, y)S_{v'_j}(x, y) \mid Q(x, y) \in \mathcal{N}^{(j-1)} \right\}, \quad (1.24)$$

```

MERGE(( $T_0(x, y), \dots, T_u(x, y)$ ), ( $S_0(x, y), \dots, S_v(x, y)$ ),  $\Delta_0$ )
1  for  $i \leftarrow 0$  to  $u + v$ 
2  do  $Q_i(x, y) = \min_{0 \leq j \leq v} T_{i-j}(x, y)S_j(x, y)$ 
3   $\mathcal{B} = (Q_0(x, y), \dots, Q_{u+v}(x, y))$ 
4  while  $\Delta(\mathcal{B}) > \Delta_0$ 
5  do  $\alpha_i \leftarrow \text{rand}()$ ,  $i = 0, \dots, u$ 
6      $\beta_j \leftarrow \text{rand}()$ ,  $j = 0, \dots, v$ 
7      $Q(x, y) \leftarrow ((\sum_{i=0}^u \alpha_i T_i(x, y))(\sum_{i=0}^v \beta_i S_i(x, y)))$ 
8      $\mathcal{B} \leftarrow \text{REDUCE}(\mathcal{B}, Q(x, y))$ 
9  return  $\mathcal{B}$ 

```

Рис. 1.16. Построение базиса Грёбнера произведения идеалов

Алгоритм *Reduce* сохраняет степени по  $y$  старших членов построенных полиномов, в силу чего  $\Delta(\mathcal{B}^{(j+1)}) \leq \Delta(\mathcal{B}^{(j)})$ . Равенство  $\Delta(\mathcal{B}^{(j)}) = \frac{n(r_1+r_2)(r_1+r_2+1)}{2}$  достигается, только если  $\mathcal{B}^{(j)}$  является базисом Грёбнера идеала  $I_{r_1+r_2}$ . Заметим, что модуль  $I_{r_1+r_2, u+v+1}$  всегда может быть порожден базисом  $\mathcal{B}^{((u+1)(v+1)-1)}$ .

Можно сократить число дополнительных полиномов путем построения произведений линейных комбинаций полиномов из данных базисов. Тогда при  $j \geq u + v$  последовательность базисов задается выражением

$$\mathcal{B}'^{(j+1)} = \text{Reduce} \left( \mathcal{B}'^{(j)}, \left( \sum_{i=0}^u \alpha_{i,j} T_i(x, y) \right) \left( \sum_{i=0}^v \beta_{i,j} S_i(x, y) \right) \right),$$

где  $\alpha_{i,j}, \beta_{i,j} \in \mathbb{F}_q \setminus \{0\}$  — случайные значения. Кроме того, первоначальный базис  $\mathcal{B}'^{(u+v)} = (B_0'^{(u+v)}(x, y), \dots, B_{u+v}'^{(u+v)}(x, y))$  может быть построен как  $B_i'^{(u+v)}(x, y) = T_{i-j}(x, y)S_j(x, y)$ , где каждое  $i$  и  $j$  выбирается так, что  $\text{LT } B_i'^{(u+v)}(x, y) = a_i x^{t_i} y_i$  и  $t_i, 0 \leq i \leq u + v$ , наименьшие возможные. Такой подход позволяет снизить число итераций алгоритма *Reduce*. Алгоритм *Merge*, предназначенный для перемножения идеалов, представлен на Рис. 1.16.

Если даны базисы  $\mathcal{T} = (T_0(x, y), \dots, T_u(x, y))$  и  $\mathcal{S} = (S_0(x, y), \dots, S_v(x, y))$  идеалов  $I_{r_1}$  и  $I_{r_2}$  соответственно, то результатом выполнения алгоритма *Merge*( $\mathcal{T}, \mathcal{S}, nr(r+1)/2$ ) является базис Грёбнера идеала  $I_{r_1+r_2}$ . Двоичный интер-

```

INTERPOLATE (((xi, yi), i = 0, ..., n - 1), r)
1  φ(x) ← ∏i=0n-1 (x - xi)
2  Υ(x) ← ∑i=0n-1 yi  $\frac{\prod_{j \neq i} (x - x_j)}{\prod_{j \neq i} (x_i - x_j)}$ 
3  G ← (φ(x))
4  repeat
5      G ← REDUCE(G, yi(y - Υ(x)))
6      j ← j + 1
7  until LT(Gj) = yj
8  B ← G
9  Let r = ∑j=0h rj2j, rj ∈ {0, 1}
10 R ← 1
11 for j ← h - 1 to 0
12 do R ← 2R
13   B ← MERGE(B, B, nR(R + 1)/2)
14   if rj = 1
15       then R ← R + 1
16       B ← MERGE(B, G, nR(R + 1)/2)
17 return B

```

Рис. 1.17. Построение базиса Грёбнера идеала  $I_r$

поляционный алгоритм, использующий функции *Reduce* и *Merge*, представлен на Рис. 1.17. Напомним, что при вычислениях должно использоваться  $(1, k - 1)$ -взвешенное лексикографическое упорядочение. Из базиса, возвращаемого алгоритмом *Interpolate*, для факторизации выбирается полином с наименьшей взвешенной степенью.

Сложность двоичного интерполяционного алгоритма может быть оценена как  $O(nr^3(a \log(r\sqrt{n/k}) \log(nr) + b(n - \sqrt{nk})))$ .

```

LEE O'SULLIVAN ALGORITHM( $M, \rho$ )
1  for  $s \leftarrow 0$  to  $\rho - 1$ 
2  do  $B_s \leftarrow \prod_{0 \leq i \leq s} (y - h^{(i)}) \prod_{j \in L} (x - x_j)^{\omega_j}$ 
3   $j \leftarrow 0$ 
4  while  $j < \rho$ 
5  do  $j \leftarrow j + 1$ 
6     $s \leftarrow \text{ydeg}(B_j)$ 
7    if  $s = j$ 
8      then break
9    repeat
10       $d \leftarrow \text{xdeg}(B_j) - \text{xdeg}(B_s)$ 
11       $c \leftarrow \text{LC}(B_j) \text{LC}(B_s)^{-1}$ 
12      if  $d \geq 0$ 
13        then  $B_j \leftarrow B_j - cx^d B_s$ 
14        else  $B_s \leftarrow B_j$ 
15           $B_j \leftarrow x^{-d} B_j - c B_s$ 
16      until  $s = j$ 
17   $i \leftarrow \arg(\min_{0 \leq i < \rho} \text{LT}(B_i))$ 
18  return  $B_i$ 

```

Рис. 1.18. Интерполяционный алгоритм Ли и О'Салливана

### 1.5.2.3. Алгоритм Ли-О'Салливана

Рассмотрим алгоритм построения базиса Грёбнера идеала  $I_{M,\rho}$ , предложенный в [47]. Построим последовательность  $n \times q$  матриц  $M_s$ ,  $0 \leq s < \rho$ , и соответствующих полиномов, при этом  $M_0 = M$ . Пусть матрица  $M_s$  состоит из элементов  $\mu_{i,j}$ ,  $0 \leq i < n$ ,  $0 \leq j < q$ . Из каждой строки матрицы  $M_s$  выберем максимальный элемент

$$\mu_i = \max_{0 \leq j \leq q-1} \mu_{i,j}, \quad 0 \leq i < n.$$



Пусть  $L = \{0 \leq i < n \mid \mu_i \geq 1\}$ . Для каждого  $i \in L$  определим  $\beta_i$  такое, что  $\mu_i = \mu_{i,\beta_i}$ . Вычислим

$$B_s = \prod_{0 \leq i < s} (y - h^{(i)}) \prod_{j \in L} (x - x_j)^{\mu_j}, \quad (1.25)$$

$$h^{(s)} = \sum_{i \in L} y_{\beta_i} h_i,$$

$$h_i = \prod_{j \in L, j \neq i} \frac{x - x_j}{x_i - x_j}$$

$$M_{s+1} = \{\mu'_{i,j} \mid 0 \leq i \leq n-1, 0 \leq j \leq q-1\},$$

где  $\mu'_{i,j} = \mu_{i,i}$  для  $i \notin L$  и  $\mu'_{i,j} = \mu_{i,j} - 1$  для  $i \in L$ . Выполним данные вычисления для  $0 \leq s < \rho$ .

Заметим, что  $(y - h^{(s)})$  является кривой, проходящей через точки  $(x_i, y_{\beta_i})$  с кратностью один для каждого  $i$  из  $L$ , и что если  $L$  — пустое множество, то  $h^{(s)} = 0$  и  $B_{s+1} = yB_s$ . Для каждого  $\rho \geq 0$  можно построить  $I_{M,\rho} = \langle B_0, B_1, \dots, B_{\rho-1} \rangle$ . При этом  $\text{udeg}(B_s) = s$  для  $s \geq 0$ .

Интерполяционный алгоритм представлен на Рис. 1.18. Функция  $\text{LC}(Q(x, y))$  возвращает коэффициент при старшем члене полинома  $Q(x, y)$ , также используется обозначение  $\text{xdeg } Q(x, y) = \text{wdeg}_{(1,0)} LT(Q(x, y))$

Сложность данного алгоритма можно оценить как  $O(n^2 m \rho^4)$ , где величина  $m$  — наибольший элемент матрицы кратностей корней.

### 1.5.3. Метод перекодирования

При декодировании методом Кёттера-Варди наиболее трудоемким является интерполяционный этап, сложность которого может быть снижена путем применения метода перекодирования, предложенного в [26].

#### 1.5.3.1. Замена переменных

Метод перекодирования предполагает замену переменных, для осуществления которой необходимо выполнить следующие шаги:

- Вычисление наиболее вероятного жесткого решения  $\bar{v} = (v_0, \dots, v_{n-1})$  с  $v_i = y_{\beta_i}$ , где  $\beta_i = \arg \max_j \{M_{i,j}\}$  для  $0 \leq i < n$ . Обозначим кратности интерполяционных точек  $(x_i, v_i)$  как  $\mu_i = M_{i,\beta_i}$ ,  $0 \leq i < n$ .
- Построение интерполяционного полинома  $f(x)$  по  $k$  точкам  $(x_i, v_i)$  с наибольшими кратностями  $\mu_i$ , такие точки в дальнейшем будут называться локаторами перекодирования. Таким образом,  $\deg f(x) < k$  и соответствующее кодовое слово  $\bar{c} = (c_0, \dots, c_{n-1})$ , при  $c_i = f(x_i)$ , совпадает с  $\bar{v}$  не менее чем в  $k$  позициях.
- Преобразование второй компоненты интерполяционных точек:  $y_i \leftarrow y_i - c_i$ .

После выполнения данных шагов интерполяционные точки могут быть разделены на три множества:  $R$ ,  $V$  и  $E$ , состоящих из точек  $(x_i, y_j)$ ,  $(x_{\hat{i}}, y_{\hat{j}})$  и  $(x_{\tilde{i}}, y_{\tilde{j}})$ , соответственно. Множество  $R$  — множество локаторов перекодирования,  $V$  — множество точек  $(x_{\hat{i}}, y_{\hat{j}})$  с  $\hat{i} \notin \{R\}_x$ , и  $E$  — множество точек  $(x_{\tilde{i}}, y_{\tilde{j}}) \notin R$  с  $\tilde{i} \in \{R\}_x$ . Таким образом, эти три множества интерполяционных точек обладают следующими свойствами:

$$\forall (x_i, y_j) \in R \quad y_j = 0,$$

$$\forall (x_i, y_j) \in R \quad \forall (x_{\hat{i}}, y_{\hat{j}}) \in V \quad M_{i,j} \geq M_{\hat{i},\hat{j}},$$

$$\forall (x_i, y_j) \in R \quad \forall (x_{\tilde{i}}, y_{\tilde{j}}) \in E \quad M_{i,j} \geq M_{\tilde{i},\tilde{j}}.$$

Как показано в [26], если полином  $Q(x, y)$  с  $\text{wdeg}_{(0,1)} Q(x, y) = \gamma$  проходит через все точки из множеств  $R$ ,  $V$  и  $E$  с заданными кратностями  $M_{i,j}$ , то  $Q(x, y)$  может быть представлен как

$$Q(x, y) = \sum_{t=0}^{\gamma} Q_t(x) \prod_{(x_i, y_j) \in R} (x - x_i)^{\max(M_{i,j}-t, 0)} y^t. \quad (1.26)$$

Пусть  $\Upsilon_t(x) = \prod_{(x_i, y_j) \in R} (x - x_i)^{\max(t-M_{i,j}, 0)}$ , тогда

$$Q(x, y) = \sum_{t=0}^{\gamma} Q_t(x) \prod_{(x_i, y_j) \in R} (x - x_i)^{M_{i,j}-t} \Upsilon_t(x) y^t.$$

Множители  $(x - x_i)^{M_{i,j}}$  могут быть вынесены за скобку и исключены из рассмотрения. Сложность вычислений может быть снижена путем перехода от множеств  $V$  и  $E$  к

$$\widehat{V} = \left\{ (x_{\widehat{i}}, z_{\widehat{j}}) \mid z_{\widehat{j}} = \frac{y_{\widehat{j}}}{\psi(x_{\widehat{i}})}, (x_{\widehat{i}}, y_{\widehat{j}}) \in V \right\},$$

$$\widehat{E} = \left\{ (x_{\widetilde{i}}, z_{\widetilde{j}}) \mid z_{\widetilde{j}} = \frac{y_{\widetilde{j}}}{\psi'(x_{\widetilde{i}})}, (x_{\widetilde{i}}, y_{\widetilde{j}}) \in E \right\},$$

соответственно, что ведет к исключению из рассмотрения точек из  $R$ , где  $\psi(x) = \prod_{i \in \{R\}_x} (x - x_i)$  и  $\psi'(x)$  — первая формальная производная функции  $\psi(x)$ .

### 1.5.3.2. Интерполяция

Задача интерполяции по множествам точек  $\widehat{V}$  и  $\widehat{E}$  состоит в построении последовательности полиномов  $\{Q_t(x)\}_{0 \leq t \leq \gamma}$ , удовлетворяющих следующим условиям:

- Полином  $\Phi(x, z) = \sum_{t=0}^{\gamma} Q_t(x) \Upsilon_t(x) z^t$  проходит через каждую интерполяционную точку  $(x_{\widehat{i}}, z_{\widehat{j}}) \in \widehat{V}$  с кратностью  $M_{\widehat{i}, \widehat{j}}$ .
- Полином  $\Theta^{\widetilde{i}}(x, z) = \sum_{t=0}^{\gamma} Q_t(x) (x - x_{\widetilde{i}})^{M_{\widetilde{i}, \widetilde{j}} - t} \Upsilon_t(x) z^t$  проходит через каждую интерполяционную точку  $(x_{\widetilde{i}}, z_{\widetilde{j}}) \in \widehat{E}$  с кратностью  $M_{\widetilde{i}, \widetilde{j}}$ .
- Полином  $\Phi(x, z)$  имеет наименьшую возможную  $(1, -1)$ -взвешенную степень.

Для заданной последовательности полиномов  $\{Q_t(x)\}_{0 \leq t \leq \gamma}$  искомый интерполяционный полином  $Q(x, y)$  восстанавливается согласно выражению (1.26).

### 1.5.4. Метод Чейза

Метод, предложенный Чейзом в [18], может быть использован при декодировании произвольного линейного блочного кода. Данный метод предполагает

перебор возможных значений для  $D$  наименее надежных символов и исправление ошибок в прочих символах с помощью алгебраического алгоритма декодирования. Число  $D$  является параметром алгоритма. В работе [16] была предложена эффективная реализация метода Чейза для случая кодов Рида-Соломона, согласно которой для каждого из  $D$  наименее надежных символов рассматривается по два наиболее вероятных значения, также обеспечивается многократное переиспользование результатов промежуточных вычислений жесткого алгебраического декодера, исправляющего не более  $(n - k)/2$  ошибок в метрике Хэмминга.

## 1.6. Выводы по разделу

В данном разделе были рассмотрены полярные коды с двоичным  $l \times l$  ядром  $M$ , то есть двоичные линейные блочные коды, порождаемые несколькими строками матрицы  $M^{\otimes m}$ . Особое внимание уделено случаям ядра Арикана  $A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$  и ядра БЧХ, то есть ядра построенного на базе расширенных кодов БЧХ. Введено понятие классических полярных кодов как полярных кодов, порождаемых строками матрицы поляризующего преобразования, которым соответствуют битовые подканалы с наименьшими вероятностями ошибки. Приведен обзор методов декодирования полярных кодов и кодов Рида-Соломона. Классические полярные коды обеспечивают наименьшую возможную вероятность ошибки декодирования методом последовательного исключения среди полярных кодов для заданного канала. Было показано, что алгоритм последовательного исключения обладает малой сложностью  $O(n \log_l(n))$ , при этом его корректирующая способность также низка. Списочный/стековый алгоритм последовательного исключения при достаточном размере списка/стека  $L$  способен обеспечить декодирование по максимуму правдоподобия, однако его вычислительная сложность  $O(Ln \log_l(n))$  при этом оказывается слишком высокой для

практического применения. Актуальной задачей является построение эффективных алгоритмов декодирования полярных кодов, обладающих малой сложностью.

Даже при декодировании по максимуму правдоподобия корректирующая способность классических полярных кодов с ядром Арикана при небольших длинах  $n$  оказывается меньше, чем у других распространенных кодов с аналогичными параметрами, к примеру, кодов МППЧ (см. раздел 1.3.2). Это обусловлено малым минимальным расстоянием полярных кодов. Таким образом, возникает необходимость разработки конструкций полярных кодов с увеличенным минимальным расстоянием.

В [44] было показано, что  $l \times l$  ядра поляризации при  $l > 2$  могут обладать скоростью поляризации большей, чем ядро Арикана. Однако задача построения полярных кодов с произвольным ядром поляризации остается открытой.

Длина полярных кодов с  $l \times l$  ядром имеет вид  $n = l^m$ , однако для различных практических приложений могут потребоваться коды с большей градацией длин. Для таких приложений необходимо построить коды произвольных длин на базе полярных кодов. При этом для кодирования и декодирования этих кодов должны быть применимы методы, разработанные для полярных кодов.

Одним из наиболее широко используемых классов корректирующих кодов являются коды Рида-Соломона. Для мягкого декодирования кодов Рида-Соломона может быть применен метод Кёттера-Варди, наиболее трудоемким шагом которого является построение полинома от двух переменных, имеющего корни различной кратности. Для реализации этого шага был предложен ряд алгоритмов двумерной интерполяции, несмотря на это, сложность метода Кёттера-Варди остается достаточно высокой. Следует отметить, что метод Кёттера-Варди не обеспечивает декодирование по максимуму правдоподобия, поэтому актуальной является задача построения для кодов Рида-Соломона алгоритмов декодирования с большей корректирующей способностью.

## Глава 2

# Построение полярных кодов

### 2.1. Полярные подкоды

Классический полярный  $(n = 2^m, k)$  код  $\widehat{C}$ , то есть полярный код построенный посредством заморозки битовых подканалов с наибольшими вероятностями ошибки, обеспечивает наименьшую возможную вероятность ошибки при декодировании методом последовательного исключения. Однако могут существовать  $(n, k)$  линейные коды, обеспечивающие меньшую вероятность ошибки декодирования при использовании списочного/стекового алгоритма последовательного исключения, чем полярный код  $\widehat{C}$ . Все рассматриваемые в данном разделе полярные коды являются полярными кодами с ядром Арикана.

#### 2.1.1. Представление линейных кодов для декодирования методом последовательного исключения

В данном разделе показано, как может быть выполнено декодирование произвольного линейного блочного кода длины  $2^m$  методом последовательного исключения или его аналогами.

Рассмотрим  $n \times n$  матрицу поляризующего преобразования  $G_n = \Lambda^{(2,m)} A^{\otimes m}$ , где  $n = 2^m$ , ядро Арикана  $A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$  и перестановочная матрица  $\Lambda^{(2,m)}$  построена как в разделе 1.1. Поскольку матрица  $G_n$  является обратимой, любой  $(n = 2^m, k, d)$  линейный код  $C$  над полем  $\mathbb{F}_q$  может быть представлен в виде подпространства пространства строк матрицы  $G_n$ , а именно, кодовое слово  $c \in C$  можно представить как  $c_0^{n-1} = u_0^{n-1} G_n$ ,  $u_0^{n-1} \in \mathbb{F}_q^n$ . Пусть  $H$  – проверочная

матрица кода  $C$ , тогда последовательность  $u_0^{n-1}$  удовлетворяет условию

$$u_0^{n-1} \underbrace{G_n H^T}_{V^T} = 0.$$

Выполняя линейные операции над строками матрицы  $H$ , построим матрицу  $V$  такую, что  $V_{j,i_j} = -1$  и для любого  $t \in \{0, \dots, 2^m - 1\}$  существует не более одного  $j : i_j = t$ , где  $i_j = \max \{t \in \{0, \dots, n-1\} | V_{j,t} \neq 0\}$ ,  $0 \leq j < n - k$ . Если рассматривать  $u_0^{n-1} V^T = 0$  как систему из  $n - k$  линейных уравнений относительно элементов последовательности  $u_0^{n-1} \in \mathbb{F}_q^n$ , то все решения данной системы можно представить как последовательности, состоящие из  $k$  произвольных элементов  $u_t \in \mathbb{F}_q$  при  $t \in \mathcal{N}$ , где  $\mathcal{N} = \{0, \dots, n-1\} \setminus \mathcal{F}$  при  $\mathcal{F} = \{i_j | 0 \leq j < n - k\}$ , и  $n - k$  зависящих от них элементов

$$u_i = \sum_{t=0}^{i-1} Z_{i,t} u_t, i \in \mathcal{F}, \quad (2.1)$$

где  $Z_{i,t} = V_{j,t}$  при  $0 \leq j < n - k$  и  $0 \leq t < n$ .

Символы  $u_i, i \in \mathcal{F}$ , мы будем называть динамически замороженными, а прочие  $k$  символов  $u_i$  — информационными символами. Таким образом, множество  $\mathcal{F}$  содержит индексы динамически замороженных символов. Если для некоторого  $i \in \mathcal{F}$  условие (2.1) вырождается в  $u_i = 0$ , то такие символы будем называть статически замороженными символами. Статически замороженные символы являются частным случаем динамически замороженных символов. Кодирование в коде  $C$  информационной последовательности  $a_0^{k-1}$  можно также представить как

$$c_0^{n-1} = a_0^{k-1} \underbrace{T}_{u_0^{n-1}} G_n,$$

где  $T$  — такая  $k \times n$  матрица, что выполняется условие  $TV^T = 0$  и столбцы матрицы  $T$  с индексами из множества  $\mathcal{N}$  составляют единичную матрицу.

Для информационной последовательности, состоящей из  $u_i \in \mathbb{F}_q, i \in \mathcal{N}$ , вычислим  $u_i, i \in \mathcal{F}$ , согласно выражению (2.1) и построим кодовое слово кода  $C$

как  $c_0^{n-1} = u_0^{n-1}G_n$ . Пусть при передаче кодового слова  $c_0^{n-1}$  по каналу была принята последовательность  $y_0^{n-1}$ . Тогда оценки  $\hat{u}_i$  символов  $u_i$  могут быть найдены с помощью метода последовательного исключения или его аналогов при условии замены выражения (1.10) на

$$\hat{u}_i = \begin{cases} \arg \max_{u_i \in \mathbb{F}_q} P(\hat{u}_0^{i-1}, u_i | y_0^{n-1}), & i \notin \mathcal{F} \\ \sum_{t=0}^{i-1} Z_{i,t} \hat{u}_t, & i \in \mathcal{F}. \end{cases} \quad (2.2)$$

Использование метода последовательного исключения предполагает последовательное принятие решений относительно значений символов  $\hat{u}_i$ . На  $i$ -ой фазе декодирования значения  $\hat{u}_t$ ,  $t < i$ , уже не могут быть пересмотрены, и значение  $\hat{u}_i$ ,  $i \in \mathcal{F}$ , определяется значениями  $\hat{u}_t$ ,  $t < i$ . На вероятности ошибки декодирования  $P_i$  символов  $u_i$ ,  $i \in \mathcal{N}$ , при условии того, что значения предыдущих символов были найдены правильно, наличие условий динамической заморозки влияния не оказывает. Таким образом, вероятность ошибки декодирования кода  $C$  методом последовательного исключения равна  $P^{(e)} = 1 - \prod_{i \in \mathcal{N}} (1 - P_i)$ , что совпадает с вероятностью ошибки декодирования полярного кода с тем же множеством индексов информационных символов  $\mathcal{N}$ . Следует отметить, что множество  $\mathcal{N}$  для произвольного линейного кода может включать большое число подканалов с большими  $P_i$ .

Если рассматриваемый код  $C$  является двоичным, то сложность декодирования методом последовательного исключения составляет  $O(n \log(n))$  операций над вещественными числами и  $n^2$  над элементами поля  $\mathbb{F}_2$ . В случае  $q$ -ичного кода  $C$  сложность декодирования методом последовательного исключения составляет  $O(n \log(n)q^2)$  операций над вещественными числами и  $n^2$  над элементами поля  $\mathbb{F}_q$ . Применение быстрого преобразования Адамара позволяет понизить число операций над вещественными числами до  $O(n \log(n)q \log(q))$ .



## 2.1.2. Полярные подкоды БЧХ

В данном разделе решается задача построения кодов, декодирование которых может быть эффективно выполнено с помощью списочного/стекового алгоритма последовательного исключения и его аналогов.

Заметим, что конструкция классических полярных кодов обеспечивает минимизацию вероятности ошибки только для случая декодирования методом последовательного исключения, что соответствует списочному/стековому алгоритму последовательного исключения с размером списка 1. При использовании стекового алгоритма последовательного исключения без ограничения на размер списка, что соответствует декодированию по максимуму правдоподобия, вероятность ошибки декодирования  $(n, k)$  классического полярного кода оказывается существенно выше принципиально достижимой  $(n, k)$  линейными блоковыми кодами при декодировании по максимуму правдоподобия. Причиной этого является малое минимальное расстояние классических полярных кодов. Как показано в [35], оно увеличивается пропорционально квадратному корню длины кода. Существует множество классов линейных блоковых кодов с существенно большим минимальным расстоянием, например, коды БЧХ.

### 2.1.2.1. Расширенные коды БЧХ

В общем случае, множество замороженных символов  $\mathcal{F}$ , построенное для произвольного кода  $C$  описанным в разделе 2.1.1 способом, включает много символов  $u_i$  с малыми  $P_i$ , в то время как многие символы с большими  $P_i$  оказываются незамороженными. Заметим, что в случае двоичного кода  $C$  вероятность  $P_i$  может также рассматриваться как вероятность ошибки в подканале  $W_{G_n}^{(i)}$ , задаваемом выражением (1.1). При этом, из выражений (1.3) и (1.4) следует, что в случае ядра Арикана  $P_i \leq Z(W_{G_n}^{(i)}) = O(Z(W)^{\text{wt}(i)})$ . Таким образом, величины  $P_i$  зависят от числа единиц в двоичной записи индексов  $i$ , следовательно декодирование методом последовательного исключения и его аналогами может быть

эффективно выполнено только для таких кодов, у которых замороженными оказываются входные символы с номерами имеющими малый вес. Примером таких кодов являются коды Рида-Маллера, а также расширенные коды БЧХ, являющиеся подкодами кодов Рида-Маллера. Действительно,  $(r, m)$  код Рида-Маллера может рассматриваться как случай полярного кода с множеством замороженных символов  $\mathcal{F} = \{i \in \{0, \dots, 2^m - 1\} \mid \text{wt}(i) < m - r\}$ . В [5, 24, 39] показано, что код, получаемый путем выкалывания одного символа из кода Рида-Маллера порядка  $r$  и длины  $2^m$ , содержит подкод, эквивалентный циклическому коду с порождающим полиномом  $g(x)$  с корнями  $\alpha^i : 1 \leq \text{wt}(i) < m - r, 1 \leq i \leq 2^m - 2$ , где  $\alpha$  — примитивный элемент поля  $\mathbb{F}_{2^m}$ . Следовательно, для заданного расширенного кода БЧХ можно построить соответствующий код Рида-Маллера такой, что все подканалы  $i$  со сравнительно малыми  $\text{wt}(i)$  являются замороженными. Однако для достижения вероятности ошибки декодирования, сопоставимой с демонстрируемой другими известными методами, приходится использовать списочный/стековый алгоритм последовательного исключения с размером списка, увеличивающимся экспоненциально с ростом длины кода. При использовании размера списка, обеспечивающего практически приемлемую сложность, вероятность ошибки декодирования длинных кодов БЧХ оказывается существенно больше вероятности ошибки их декодирования по максимуму правдоподобия.

**Пример 1.** Рассмотрим  $(16, 7, 6)$  расширенный код БЧХ. Порождающий полином соответствующего нерасширенного кода имеет корни  $\alpha, \alpha^3$  и сопряженные с ними элементы, где  $\alpha$  — примитивный корень  $x^4 + x + 1$ . Для того, чтобы последовательность  $uG_n$  являлась кодовым словом рас-

ширенного кода БЧХ, вектор  $u$  должен удовлетворять условию

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

Перемножая матрицы, раскладывая элементы по стандартному базису и применяя элементарные операции над столбцами, получаем

$$u \begin{pmatrix} 0000010000101000 \\ 0000000001100000 \\ 0000011001000000 \\ 0000000010000000 \\ 0001010000000000 \\ 0000100000000000 \\ 0010000000000000 \\ 0100000000000000 \\ 1000000000000000 \end{pmatrix}^T = 0 \quad (2.3)$$

Отсюда получаем, что  $u_0 = u_1 = u_2 = u_4 = u_8 = 0$ ,  $u_5 = u_3$ ,  $u_9 = u_5 + u_6$ ,  $u_{10} = u_9$ ,  $u_{12} = u_5 + u_{10} = u_6$ , то есть символы  $u_0, u_1, u_2, u_4$  и  $u_8$  являются статически замороженными,  $u_5, u_9, u_{10}$  и  $u_{12}$  — динамически замороженными,

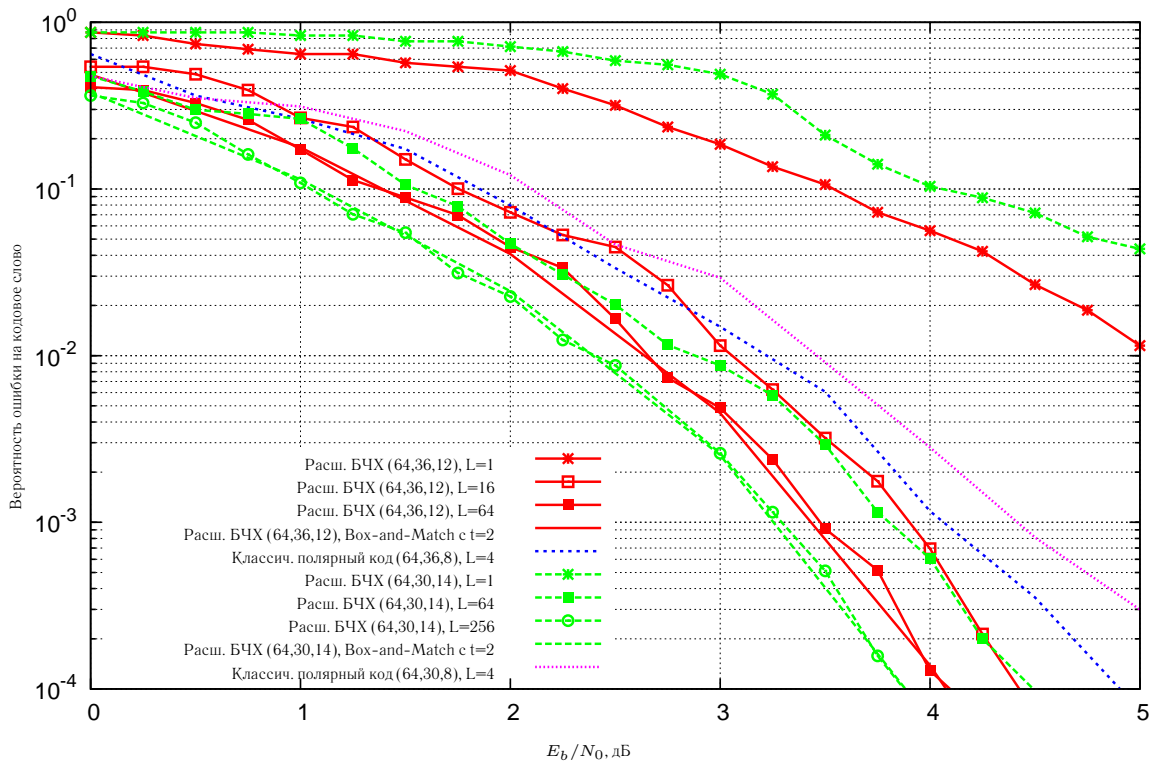


Рис. 2.1. Вероятность ошибки декодирования расширенных кодов БЧХ

$a$   $u_3, u_6, u_7, u_{11}, u_{13}, u_{14}, u_{15}$  — информационными символами (незамороженными символами).

Использование взаимосвязи между расширенными кодами БЧХ и кодами Рида-Маллера не обеспечивает включение всех плохих битовых подканалов в множество замороженных. Множество динамически замороженных подканалов низкоскоростного расширенного кода БЧХ включает много хороших подканалов, в то время как многие подканалы с высокой вероятностью ошибки остаются незамороженными. Это ведет к необходимости использования списочного алгоритма последовательного исключения с чрезвычайно большим размером списка для обеспечения вероятности ошибки декодирования, сравнимой с таковой других алгоритмов декодирования. На Рис. 2.1 представлен график зависимости вероятности ошибки декодирования от отношения сигнал/шум для расширенных кодов БЧХ и полярных кодов с  $64 \times 64$  ядром Арикана  $A^{\otimes 6}$ , при декодировании которых использовался как списочный алгоритм последовательного ис-

ключения, так и алгоритм Vox-and-Match, предложенный в [80]. Видно, что для обеспечения вероятности ошибки декодирования, сравнимой с алгоритмом Vox-and-Match, размер списка в алгоритме последовательного исключения должен быть не менее 64.

### 2.1.2.2. Предлагаемая конструкция кодов

**Определение 3.**  $(n = l^m, k)$  полярным подкодом с  $l \times l$  ядром  $M$  и с множеством дополнительно замороженных символов  $\tilde{\mathcal{F}}$ ,  $|\tilde{\mathcal{F}}| = k' - k$ , двоичного линейного  $(n, k', d)$  кода  $C'$  называется код  $C = \{u_0^{n-1} G_n \in C' | u_0^{n-1} \in \{0, 1\}^n, \forall i \in \tilde{\mathcal{F}} u_i = 0\}$ , где матрица поляризующего преобразования  $G_n = \Lambda^{(l,m)} M^{\otimes m}$ .

В дальнейшем ограничимся рассмотрением полярных подкодов с ядром Арикана.

Двоичный линейный  $(n = 2^m, k, \geq d)$  код, декодирование которого может быть эффективно выполнено с помощью списочного/стекового алгоритма последовательного исключения и его аналогов, мы предлагаем построить как полярный подкод  $(n, k', d)$  расширенного кода БЧХ, обеспечивающий минимально возможную вероятность ошибки декодирования методом последовательного исключения для заданного канала передачи данных. Такой подход позволяет получить  $(n, k)$  код  $C$ , обеспечивающий меньшую вероятность ошибки при декодировании по максимуму правдоподобия, чем  $(n, k)$  классический полярный код  $\hat{C}$ , построенный для этого же канала. Если  $k'$  существенно больше  $k$ , то вероятность ошибки декодирования методом последовательного исключения кода  $C$  лишь незначительно превосходит такую кода  $\hat{C}$ , что избавляет от необходимости использования чрезмерно большого размера списка при декодировании с помощью списочного/стекового алгоритма последовательного исключения.

$(n = 2^m, k)$  полярным подкодом БЧХ будем называть  $(n = 2^m, k)$  полярный подкод с ядром Арикана  $(n, k' \geq k, d)$  расширенного кода БЧХ  $C'$ , обеспечи-

вающий наименьшую возможную вероятность ошибки декодирования методом последовательного исключения для заданного канала передачи информации.

Предлагаемая процедура построения  $(n = 2^m, k)$  кода для списочного/стекового алгоритма последовательного исключения с заданным параметрами (например, размером списка  $L$ ) и заданного канала состоит в

1. Построении набора  $\mathcal{C}$  из  $(n, k', d)$  расширенных примитивных кодов БЧХ в узком смысле со всеми возможными размерностями  $k \leq k' \leq n$ .
2. Для каждого кода  $C' \in \mathcal{C}$ :
  - а. Выявлении множества динамически замороженных символов  $\mathcal{F}'$  согласно процедуре, описанной в разделе 2.1.1.
  - б. Построении множества дополнительно замороженных символов  $\tilde{\mathcal{F}}$  как  $k' - k$  символов  $u_i, i \in \{0, \dots, n - 1\} \setminus \mathcal{F}'$ , с наибольшими вероятностями ошибки  $P_i$  (см. раздел 1.3.1).
  - в. Нахождении оценки<sup>1</sup>  $P^{(C)}$  вероятности ошибки декодирования построенного  $(n, k, \geq d)$  полярного подкода  $C$  кода  $C'$  при использовании списочного/стекового алгоритма последовательного исключения с заданными параметрами.
3. Выборе среди построенных  $(n, k)$  полярных подкодов  $C$  подкода

$$C^* = \arg \max_C P^{(C)}.$$

Аналогичным образом могут быть построены полярные подкоды и иных линейных блочных кодов.

**Пример 2.** Построим  $(16, 6, 6)$  полярный подкод БЧХ для случая двоичного стирающего канала с вероятностью стирания  $p_{0,0} = 0.5$ , используя  $(16, 7, 6)$  расширенный код БЧХ, рассмотренный в Примере 1.

---

<sup>1</sup> Данная оценка может быть получена с помощью методов статистического моделирования.

Используя выражения (1.5) и (1.6), получаем последовательность вероятностей стирания в битовых подканалах (0.9999, 0.992, 0.985, 0.77, 0.96, 0.65, 0.53, 0.1, 0.9, 0.47, 0.35,  $3.7 \cdot 10^{-2}$ , 0.23,  $1.5 \cdot 10^{-2}$ ,  $7.8 \cdot 10^{-3}$ ,  $1.5 \cdot 10^{-5}$ ). Подчеркнутые значения соответствуют замороженным битовым подканалам расширенного кода БЧХ. Видно, что  $u_3$  имеет наибольшую вероятность стирания 0.77 среди незамороженных, следовательно, именно 3-ий подканал должен быть заморожен, чтобы получить требуемый код.

**Пример 3.** Рассмотрим построение (1024, 512) кода. Списочный декодер последовательного исключения не способен эффективно выполнить декодирование (1024, 513, 116) расширенного кода БЧХ. С другой стороны, (1024, 512) классический полярный код, оптимизированный для Гауссовского канала с отношением сигнал/шум 2 дБ, имеет минимальное расстояние 16. Построим (1024, 512,  $\geq 24$ ) полярный подкод БЧХ на базе (1024, 913, 24) расширенного кода БЧХ путем дополнения множества замороженных символов 401 символами  $u_i$ , которым соответствуют наибольшие  $P_i$ .

На Рис. 2.2 представлен график зависимости вероятности ошибки декодирования от отношения сигнал/шум для (1024, 512) полярных подкодов БЧХ, построенных на базе расширенных кодов БЧХ длины 1024 с различным минимальным расстоянием  $d$ . Кроме того, приведены данные для классического полярного кода и для полярного кода с контрольной суммой CRC [72], а также (1032, 516) кода МППЧ. Декодирование полярных кодов выполнялось с помощью списочного алгоритма последовательного исключения с размером списка  $L = 32$  и  $L = 128$ . Видно, что благодаря увеличенному минимальному расстоянию, предлагаемые полярные подкоды БЧХ обеспечивают существенно меньшую вероятность ошибки декодирования по сравнению с прочими представленными на графике кодами. Код с конструктивным минимальным расстоянием  $d = 24$  превосходит как классический полярный код с контрольной суммой CRC, так и код с

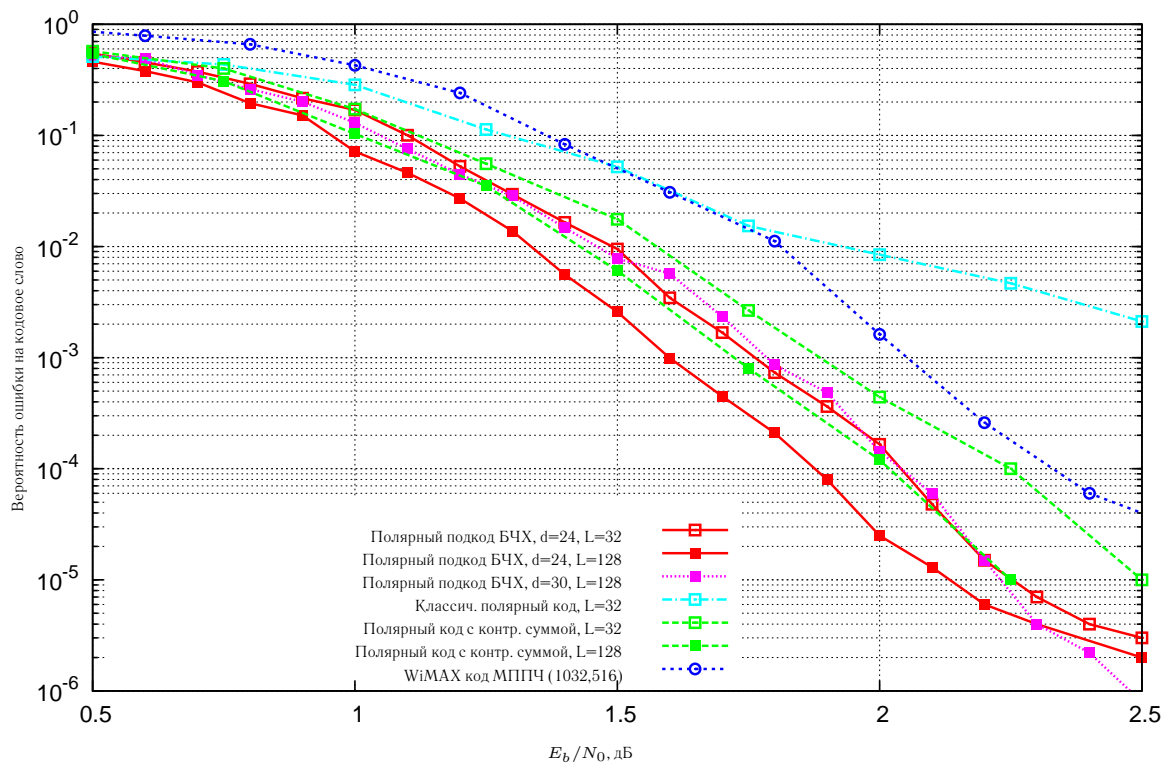


Рис. 2.2. Вероятность ошибки декодирования кодов (1024, 512)

малой плотностью проверок на четность. При увеличении минимального расстояния хорошие битовые подканалы могут оказаться замороженными, что ведет к снижению корректирующей способности списочного/стекового алгоритма последовательного исключения в области малых отношений сигнал/шум. Для того, чтобы полностью использовать корректирующую способность полярных кодов с контрольной суммой CRC и предлагаемых кодов при декодировании требуется выбрать большой размер списка  $L$ .

### 2.1.3. Выводы

В данном разделе показано, как может быть выполнено декодирование произвольного  $(n = 2^m, k)$  линейного кода методом последовательного исключения или его аналогами. При декодировании используется представление кода, при котором проверки на четность для символов кодового слова, задаваемые проверочной матрицей  $H$ , преобразуются в условия динамической заморозки (2.1)



для  $n - k$  символов входной последовательности  $u_0^{n-1}$ , прочие  $k$  символов которой могут принимать произвольные значения и рассматриваются в качестве информационных.

Была предложена конструкция полярного подкода заданного расширенного кода БЧХ, обеспечивающая минимизацию вероятности ошибки декодирования методом последовательного исключения. Был предложен метод выбора полярных подкодов БЧХ заданной размерности, обеспечивающий минимизацию вероятности ошибки их декодирования заданным алгоритмом. Приведены примеры  $(n, k)$  полярных подкодов БЧХ, обеспечивающих меньшую вероятность ошибки декодирования при использовании списочного/стекового алгоритма последовательного исключения, чем  $(n, k)$  классический полярный код с ядром Арикана. Показано, что полярные подкоды БЧХ могут выигрывать по вероятности ошибки декодирования у кодов МППЧ и полярных кодов с CRC.

## 2.2. Укорочение полярных кодов с ядром Арикана

Укорочение является стандартным методом для построения кодов различных длин из заданного кода. Пусть  $S_m$  — двоичный вектор длины  $2^m$ , показывающий позиции укороченных символов, то есть если  $S_{m,i} = 1$ , то  $i$ -ый символ является укороченным. Укорочение произвольного линейного блочного кода  $C$  длины  $N = 2^m$  на  $s = \text{wt}(S_m)$  символов состоит в выборе всех кодовых слов  $c \in C$  таких, что  $c_i = 0$  для  $i \in \text{supp}(S_m)$  и исключении из этих кодовых слов символов  $c_i$  для  $i \in \text{supp}(S_m)$ . Полученные таким образом вектора составляют  $(n = 2^m - s, k \geq K - s, d \geq D)$  линейный код. Использование различных шаблонов  $S_m$  приводит к построению кодов, вероятности ошибки декодирования которых существенно различаются.

### 2.2.1. Идея предлагаемого алгоритма

Рассмотрим построение  $(n, k)$  укороченного полярного кода  $C$ , где  $2^{m-1} < n \leq 2^m$ . Очевидный способ построения кода  $C$  состоит в укорочении на  $s = N - n$  символов  $(N = 2^m, k + s)$  полярного кода с использованием некоторого шаблона  $S_m$ ,  $\text{wt}(S_m) = s$ . Пусть  $\mathcal{F}$  — множество замороженных символов этого  $(N = 2^m, k + s)$  полярного кода и  $P(\mathcal{F}, S_m)$  — вероятность ошибки декодирования методом последовательного исключения для кода  $C$ . Пусть  $\mathcal{R}$  и  $\mathcal{D}$  — множества всех шаблонов  $S_m$  веса  $s$ , и всех множеств  $\mathcal{F} \subset \{0, \dots, N - 1\}$  мощности  $n - k$ , соответственно. Можно сформулировать задачу поиска оптимального шаблона для укорочения  $S_m^{(\mathcal{F})} = \arg \min_{S_m \in \mathcal{R}} P(\mathcal{F}, S_m)$ . Однако следствием укорочения является то, что некоторые символы кодового слова передаются по абсолютно надежному каналу, что влияет на вероятности ошибки в битовых подканалах, задаваемых поляризующим преобразованием. Таким образом, множество замороженных подканалов первоначального неукороченного кода может перестать быть оптимальным для укороченного кода. Мы предлагаем выполнять совместную оптимизацию множества  $\mathcal{F}$  и шаблона  $S_m$ :  $(\mathcal{F}^*, S_m^*) = \arg \min_{\substack{\mathcal{F} \in \mathcal{D} \\ S_m \in \mathcal{R}}} P(\mathcal{F}, S_m)$ , здесь  $\mathcal{F}^*$  и  $S_m^*$  — оптимальное множество замороженных символов и шаблон для укорочения, соответственно. Заметим, что  $S_m^* = \arg \min_{S_m \in \mathcal{R}} P(S_m)$  и  $\mathcal{F}^* = \arg \min_{\mathcal{F} \in \mathcal{D}} P(\mathcal{F}, S_m^*)$ , где  $P(S_m) = \min_{\mathcal{F} \in \mathcal{D}} P(\mathcal{F}, S_m)$ . Совместная оптимизация может быть реализована непосредственно путем перебора всех шаблонов  $S_m \in \mathcal{R}$ , и построения для каждого из них соответствующего множества замороженных символов  $\mathcal{F}$ , минимизирующего  $P(\mathcal{F}, S_m)$ . Однако сложность такого подхода чрезмерно высока, поскольку  $|\mathcal{R}| = \binom{N}{s}$ .

Далее приведено высокоуровневое описание предлагаемого алгоритма совместной оптимизации. Мы используем структуру полярных кодов для того, чтобы построить множество  $\mathcal{G}_0 \subset \mathcal{R}$ , удовлетворяющее следующему условию

$$\forall S_m \in \mathcal{R} \exists S'_m \in \mathcal{G}_0 : P(S'_m) = P(S_m).$$

Шаблон  $S_m^*$  может быть выбран среди  $S_m \in \mathcal{G}_0$ . Как будет показано ниже по тексту, многие шаблоны  $S_m$  являются эквивалентными в некотором смысле, что позволяет построить множество  $\mathcal{G}_0$ , мощность которого намного меньше  $|\mathcal{R}|$ . Кроме того, для осуществления эффективного поиска по множеству  $\mathcal{G}_0$  мы предлагаем рекурсивно разбивать его на подмножества. Данные подмножества могут быть упорядочены в виде дерева с корнем  $\mathcal{G}_0$ , листья которого задают шаблоны  $S_m \in \mathcal{G}_0$ . Другие вершины дерева представлены множествами  $\mathcal{G}_i = \bigcup_{j \in B(i)} \mathcal{G}_j$ , где  $B(i)$  — множество индексов потомков вершины  $\mathcal{G}_i$ . Для каждой вершины  $\mathcal{G}_i$  можно вычислить метрику  $P(\mathcal{G}_i)$  такую, что  $P(\mathcal{G}_i) \leq P(\mathcal{G}_j)$ ,  $j \in B(i)$ . Метрики для листьев  $S_m$  равны вероятностям ошибки декодирования  $P(S_m)$ , таким образом, метрики прочих вершин  $\mathcal{G}_i$  могут рассматриваться как оценки снизу для метрик шаблонов  $S_m$ , являющихся листьями поддерева с корнем  $\mathcal{G}_i$ . Предлагаемый алгоритм начинает работу с корня  $\mathcal{G}_0$ . На первой итерации вычисляются метрики для потомков корня  $\mathcal{G}_0$ . На каждой последующей итерации выбирается еще непосещенная вершина с наименьшим значением метрики, и для ее потомков вычисляются метрики. Шаблон, минимизирующий  $P(S_m)$ , соответствует первому достигнутому листу дерева. Далее мы будем называть данное дерево деревом укорочения. Предлагаемый метод выявления эквивалентных шаблонов и алгоритм обхода дерева укорочения будут описаны в последующих подразделах.

При необходимости построения  $(n, k)$  укороченного полярного кода с минимальным расстоянием не менее  $d$ , можно использовать предлагаемый алгоритм при наличии  $N - K$  ограничений на замороженные символы, задаваемых  $(N, K \geq k + s, d)$  полярным кодом с наибольшей возможной размерностью  $K$ . Пусть  $\tilde{\mathcal{F}}$  — множество динамически замороженных символов этого  $(N, K, d)$  кода. Для получения  $(n, k, \geq d)$  укороченного полярного кода поиск  $\mathcal{F}^*$  выполняется по множеству  $\{\mathcal{F} \in \mathcal{D} \mid \tilde{\mathcal{F}} \subset \mathcal{F}\}$ .

Заметим, ограничение  $c_j = 0$ ,  $j \in \text{supp}(S_m)$ , эквивалентно использованию

проверочной матрицы со строками, равными единичным векторам с единицами на позициях из множества  $\text{supp}(S_m)$ . Это приводит к появлению  $s = \text{wt}(S_m)$  динамически замороженных символов, как описано в разделе 2.1.

### 2.2.2. Эквивалентные шаблоны для укорочения

Для заданного  $S_m$  построим  $S_\lambda$ ,  $0 \leq \lambda < m$ , следующим образом

$$S_{\lambda,\phi} = S_{\lambda+1,2\phi} + S_{\lambda+1,2\phi+1}, 0 \leq \phi < 2^\lambda. \quad (2.4)$$

Следовательно,  $S_{0,0} = \text{wt}(S_m)$ . Также определим подмассивы  $S^{(\lambda,\phi)}$  массива  $S$  как

$$S_{t,i}^{(\lambda,\phi)} = S_{\lambda+t,\phi 2^t+i}, 0 \leq t \leq m - \lambda, 0 \leq i < 2^t.$$

Мы будем называть массивы  $S$  и  $\tilde{S}$ , построенные для  $2^m \times 2^m$  поляризующего преобразования, эквивалентными, если для этих массивов значения  $p_{0,i}$ , задаваемые выражением (1.18),  $0 \leq i < 2^m$ , совпадают. Можно показать, что из эквивалентности  $S$  и  $\tilde{S}$  следует  $P(S_m) = P(\tilde{S}_m)$ .

**Утверждение 1.** Пусть  $\tilde{S}$  — массив, полученный из  $S$  путем перестановки  $S^{(\lambda+1,2\phi)}$  и  $S^{(\lambda+1,2\phi+1)}$ , то есть  $\tilde{S}^{(\lambda+1,2\phi)} = S^{(\lambda+1,2\phi+1)}$  и  $\tilde{S}^{(\lambda+1,2\phi+1)} = S^{(\lambda+1,2\phi)}$ , для любых  $0 \leq \lambda < m$ ,  $0 \leq \phi < 2^\lambda$ . Тогда массив  $S$  эквивалентен массиву  $\tilde{S}$ .

*Доказательство.* Рассмотрим массив  $S^{(\lambda,\phi)}$  и вычислим для него вероятности  $p_{0,i}$ ,  $0 \leq i < 2^{m-\lambda}$ . Пусть  $i' = i \bmod \eta$ ,  $\eta = 2^{m-\lambda-1}$ . Согласно выражению (1.18) значение  $p_{0,i}$  зависит только от  $p_{1,i'}$  и  $p_{1,i'+\eta}$ . Заметим, что вероятность  $p_{1,i'+b\eta}$ , вычисленная для  $S^{(\lambda,\phi)}$ , равна вероятности  $p_{0,i'}$ , вычисленной для  $S^{(\lambda+1,2\phi+b)}$ , где  $b \in \{0, 1\}$ . Значения обеих функций  $\Psi^{(0)}$  и  $\Psi^{(1)}$ , задаваемых выражениями (1.19) и (1.20), не зависят от порядка следования их аргументов, следовательно перестановка  $S^{(\lambda+1,2\phi)}$  и  $S^{(\lambda+1,2\phi+1)}$ , ведущая к перестановке  $p_{1,i'}$  и  $p_{1,i'+\eta}$ , не влияет на значение  $p_{0,i}$ .  $\square$

Таким образом, для исключения из рассмотрения эквивалентных массивов мы предлагаем выполнять поиск только по  $S$ , удовлетворяющим

$$S_{\lambda+1,2\phi} \leq S_{\lambda+1,2\phi+1}, 0 \leq \lambda < m, 0 \leq \phi < 2^\lambda. \quad (2.5)$$

Если для  $S$ , удовлетворяющего (2.5) существуют  $\lambda$  и  $\phi$  такие, что  $S_{\lambda+1,2\phi} = S_{\lambda+1,2\phi+1}$  и  $S^{(\lambda+1,2\phi)} \neq S^{(\lambda+1,2\phi+1)}$ , то  $\tilde{S}$ , полученный из  $S$  путем перестановки  $S^{(\lambda+1,2\phi)}$  и  $S^{(\lambda+1,2\phi+1)}$ , также удовлетворяет (2.5). Такие  $S$  и  $\tilde{S}$  эквивалентны и удовлетворяют (2.5). Для того, чтобы исключить из рассмотрения один из этих массивов, мы предлагаем для таких  $S$ ,  $\lambda$  и  $\phi$  потребовать выполнения следующего дополнительного условия:

$$S_{t,i}^{(\lambda+1,2\phi)} < S_{t,i}^{(\lambda+1,2\phi+1)}, \quad (2.6)$$

где  $t = \min \left\{ t' \mid S_{t',i'}^{(\lambda+1,2\phi)} \neq S_{t',i'}^{(\lambda+1,2\phi+1)}, 0 \leq i' < 2^{t'} \right\}$  и  $i = \min \left\{ i' \mid S_{t,i'}^{(\lambda+1,2\phi)} \neq S_{t,i'}^{(\lambda+1,2\phi+1)} \right\}$ .

Пусть  $\mathcal{M}^{(N)}(s)$  — количество массивов  $S$ , удовлетворяющих условиям (2.5) и (2.6). Для нечетных  $s$  получаем  $S^{(1,0)} \neq S^{(1,1)}$ , следовательно

$$\mathcal{M}^{(N)}(s) = \sum_{i=\max(0,s-N/2)}^{\lfloor s/2 \rfloor} \mathcal{M}^{(N/2)}(i) \mathcal{M}^{(N/2)}(s-i), \quad s \in O,$$

где  $\mathcal{M}^{(1)}(0) = 1$  и  $\mathcal{M}^{(1)}(1) = 1$ . Для четных  $s$  существует  $\mathcal{M}^{(N/2)}(s/2)$  массивов  $S$ , удовлетворяющих (2.5) и (2.6) таких, что  $S^{(1,0)} = S^{(1,1)}$ . Для  $S^{(1,0)}$  и  $S^{(1,1)}$ , удовлетворяющих (2.5) и (2.6), существует  $\left( \mathcal{M}^{(N/2)}(s/2) \right)^2 - \mathcal{M}^{(N/2)}(s/2)$  массивов  $S$  таких, что  $S_{1,0} = S_{1,1}$  и  $S^{(1,0)} \neq S^{(1,1)}$ , только половина таких  $S$  удовлетворяет (2.5) и (2.6). Отсюда получаем, что

$$\begin{aligned} \mathcal{M}^{(N)}(s) &= \sum_{i=\max(0,s-N/2)}^{\lfloor s/2 \rfloor} \mathcal{M}^{(N/2)}(i) \mathcal{M}^{(N/2)}(s-i) - \\ &- \mathcal{M}^{(N/2)}(s/2) (\mathcal{M}^{(N/2)}(s/2) - 1) / 2, \quad s \in E. \end{aligned}$$

Таким образом, ограничения, описанные выше, уменьшают размер пространства поиска с  $\binom{N}{s}$  до  $\mathcal{M}^{(N)}(s)$ .

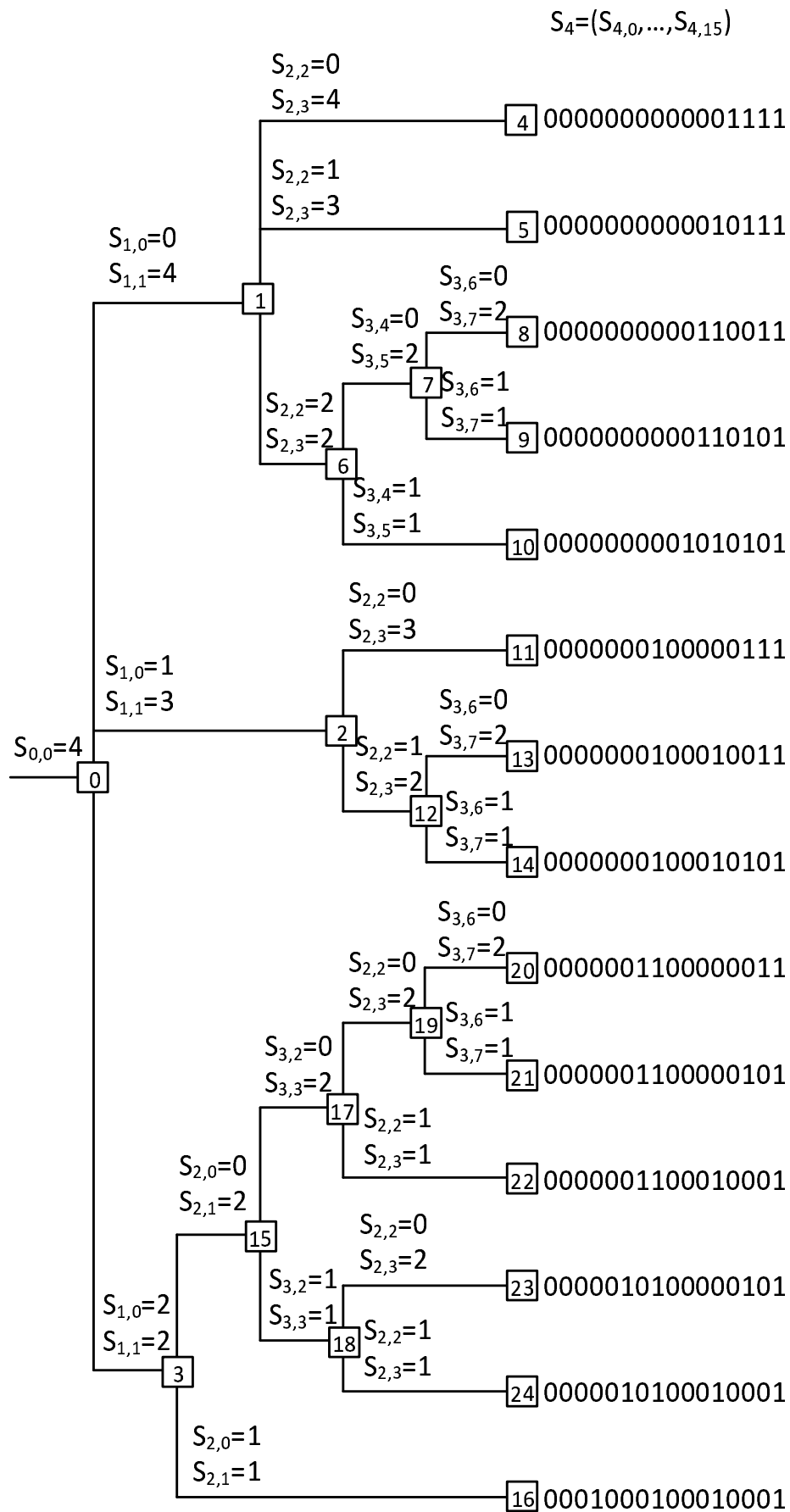


Рис. 2.3. Дерево укорочения

**Пример 4.** На Рис. 2.3 представлено дерево укорочения, построенное для  $16 \times 16$  поляризующего преобразования при параметре  $s = 4$ . Число неэквивалентных шаблонов  $S_4$  равно 14, в то время как  $\binom{16}{4} = 1820$ . Потомкам каждой вершины соответствуют различные значения некоторого  $S_{\lambda,\phi}$ . К примеру, у вершины 1 есть потомки 4, 5, 6, каждому из которых соответствует свое значение  $S_{2,2}$ , а также  $S_{2,3} = S_{1,1} - S_{2,2}$ . Множество  $\mathcal{G}_1$  состоит из шаблонов, соответствующих вершинам 4, 5, 8, 9 и 10. На Рис. 2.3 указаны только те значения  $S_{\lambda,\phi}$ , которые уникальны для соответствующих ветвей в дереве укорочения.

### 2.2.3. Алгоритм оптимизации

Предлагаемый оптимизационный алгоритм выполняет поиск по дереву укорочения, в котором каждая вершина  $\mathcal{G}_r$  представлена соответствующим массивом  $\hat{S}$  таким, что  $\hat{S}_{t,i} = S_{t,i}$  для всех  $S$ , для которых  $S_m \in \mathcal{G}_r$ , и пары  $(t, i)$ , удовлетворяющей  $(t < \lambda + 1) \vee ((t = \lambda + 1) \wedge (i < 2\phi))$ , и  $\hat{S}_{t,i} = null$  для прочих  $(t, i)$ , где  $\lambda = \min \left\{ \lambda' | S_{\lambda'+1, 2\phi'} \neq \tilde{S}_{\lambda'+1, 2\phi'}, S, 0 \leq i' < 2\phi', \tilde{S} \in \mathcal{G}_r \right\}$  и  $\phi = \min \left\{ \phi' | S_{\lambda+1, 2\phi'} \neq \tilde{S}_{\lambda+1, 2\phi'}, S, \tilde{S} \in \mathcal{G}_r \right\}$ . Массивы  $\hat{S}$  хранятся в приоритетной очереди  $Q$  вместе с соответствующими метриками  $p$ , номерами слоев  $\lambda$  и фазами  $\phi$ . Далее кортежи  $(p, \lambda, \phi, \hat{S})$  мы будем называть путями. В приоритетной очереди  $Q$  пути упорядочены согласно их метрикам  $p$ , являющимися оценками снизу для вероятности ошибки декодирования методом последовательного исключения. Слой  $\lambda$  и фаза  $\phi$  указывают позицию первого непроинициализированного элемента  $\hat{S}_{\lambda+1, 2\phi}$ .

Предлагаемый алгоритм для совместной оптимизации представлен на Рис. 2.4. Его аргументами являются длина  $n$  и размерность  $k$  укороченного полярного кода, который требуется построить, а также вычисленное заранее множество индексов статически/динамически замороженных символов  $\tilde{\mathcal{F}}$  и и соответствующие ограничения для замороженных символов  $\tilde{Z}$  (см. раздел 2.1), и ограничение

```

JOINTOPTIMIZATION( $n, k, \tilde{\mathcal{F}}, \tilde{Z}, q$ )
1   $m \leftarrow \lceil \log(n) \rceil$ 
2   $N \leftarrow 2^m$ 
3   $s \leftarrow N - n$ 
4   $\hat{S}_{0,0} \leftarrow s$ 
5   $Q \leftarrow (0, 0, 0, \hat{S})$ 
6  while SIZE( $Q$ ) > 0
7  do  $(p, \lambda, \phi, \hat{S}) \leftarrow \text{POPMIN}(Q)$ 
8      if  $\lambda = m$ 
9          then  $(\mathcal{F}, Z) \leftarrow \text{FREEZING}(n, k, \tilde{\mathcal{F}}, \tilde{Z}, \hat{S})$ 
10             return  $(\mathcal{F}, Z, \hat{S})$ 
11       $t \leftarrow \max(0, \hat{S}_{\lambda,\phi} - 2^{m-\lambda-1})$ 
12      for  $\hat{S}_{\lambda+1,2\phi} = t$  to  $\lfloor \hat{S}_{\lambda,\phi}/2 \rfloor$ 
13          do  $\hat{S}_{\lambda+1,2\phi+1} \leftarrow \hat{S}_{\lambda,\phi} - \hat{S}_{\lambda+1,2\phi}$ 
14             if NOTREDUNDANT( $\hat{S}$ )
15                 then  $p' \leftarrow \text{LOWERBOUND}(n, k, \lambda, \phi, \tilde{\mathcal{F}}, \hat{S})$ 
16                 if  $p' \neq \text{null}$ 
17                     then if (SIZE( $Q$ ) =  $q$ )
18                         then POPMAX( $Q$ )
19                         if  $\phi = 2^\lambda - 1$ 
20                             then PUSH( $Q, (p', \lambda + 1, 0, \hat{S})$ )
21                             else PUSH( $Q, (p', \lambda, \phi + 1, \hat{S})$ )
22 return null

```

Рис. 2.4. Поиск оптимального шаблона для укорочения и ограничений для замороженных символов



на размер приоритетной очереди  $q$ .

В алгоритме используются функции  $Push(Q, U)$ ,  $PopMin(Q)$  и  $PopMax(Q)$ , предназначенные для добавления в приоритетную очередь  $Q$  пути  $U$ , извлечения из  $Q$  пути с наименьшей и наибольшей метрикой, соответственно.

Изначально приоритетная очередь  $Q$  содержит только один путь ( $p = 0, \lambda = 0, \phi = 0, \hat{S}$ ),  $\hat{S}_{0,0} = s$ . Заметим, что  $p$  можно проинициализировать любым значением. На каждой итерации цикла *WHILE* из очереди  $Q$  извлекается путь  $(p, \lambda, \phi, \hat{S})$  с наименьшей  $p$ . Если  $\lambda = m$ , то  $\hat{S}_m$  является оптимальным шаблоном для укорочения, и он передается в функцию *Freezing* для построения соответствующего кода. В противном случае, в цикле *FOR* рассматриваются все возможные значения для  $\hat{S}_{\lambda+1,2\phi}$  и  $\hat{S}_{\lambda+1,2\phi+1} = \hat{S}_{\lambda,\phi} - \hat{S}_{\lambda+1,2\phi}$ , удовлетворяющие условию (2.5). Функция  $NotRedundant(\hat{S})$  возвращает *true*, если  $\hat{S}$  удовлетворяет условию (2.6), и *false*, в противном случае. Во избежании переполнения приоритетной очереди, пути с наибольшими метриками удаляются. Путь с обновленным массивом  $\hat{S}$  и соответствующей метрикой, вычисленной с помощью функции *LowerBound*, добавляется в очередь. Заметим, что инициализация массива  $\hat{S}$  выполняется по строкам.

Функция  $Freezing(n, k, \tilde{\mathcal{F}}, \tilde{Z}, \hat{S})$ , приведенная на Рис. 2.5, предназначена для построения для заданного шаблона  $\hat{S}_m$  множества замороженных символов  $\mathcal{F}$ , а также соответствующих ограничений  $Z$ , описанных в разделе 2.1. Множество  $\mathcal{F}$  должно включать множество  $\tilde{\mathcal{F}}$  и множество замороженных символов  $\hat{\mathcal{F}}$ , вычисленное функцией *FrozenSymbols* и соответствующее ограничениям  $c_i = 0, i \in \text{supp}(\hat{S}_m)$ . Подмножество  $\overline{\mathcal{F}} = \mathcal{F} \setminus (\tilde{\mathcal{F}} \cup \hat{\mathcal{F}})$  соответствует битовым подканалам с индексами  $i \in \{0, \dots, 2^m - 1\} \setminus (\tilde{\mathcal{F}} \cup \hat{\mathcal{F}})$  и наибольшими вероятностями ошибки. Ограничения  $Z$  вычисляются с использованием матрицы  $M$ , строками которой являются строки матриц  $\hat{M}$ ,  $\tilde{M}$  и  $\overline{M}$ , которые задают множества  $\hat{\mathcal{F}}$ ,  $\tilde{\mathcal{F}}$  и  $\overline{\mathcal{F}}$ , соответственно. Символы из множества  $\overline{\mathcal{F}}$  являются статически замороженными, следовательно, каждая строка матрицы  $\overline{M}$  содержит только одну едини-

```

FREEZING( $n, k, \tilde{\mathcal{F}}, \tilde{Z}, \hat{S}$ )
1  ( $\mathcal{F}, \hat{\mathcal{F}}$ )  $\leftarrow$  FROZENSYMBOLS( $n, k, \tilde{\mathcal{F}}, \hat{S}$ )
2   $m \leftarrow \lceil \log(n) \rceil$ 
3   $H \leftarrow \mathbf{0}_{|\hat{\mathcal{F}}| \times 2^m}$ 
4   $i \leftarrow 0$ 
5  for  $j \in \hat{\mathcal{F}}$ 
6  do  $H_{i,j} \leftarrow 1$ 
7     $i \leftarrow i + 1$ 
8   $\widehat{M}^T \leftarrow A^{\otimes m} H^T$ 
9   $\widetilde{M} \leftarrow \mathbf{0}_{|\tilde{\mathcal{F}}| \times 2^m}$ 
10  $\widetilde{M}_{i,j} \leftarrow 1, \quad j \in \tilde{Z}_i, 0 \leq i < |\tilde{\mathcal{F}}|$ 
11  $\overline{\mathcal{F}} \leftarrow \mathcal{F} \setminus (\tilde{\mathcal{F}} \cup \hat{\mathcal{F}})$ 
12  $\overline{M} \leftarrow \mathbf{0}_{|\overline{\mathcal{F}}| \times 2^m}$ 
13  $i \leftarrow 0$ 
14 for  $j \in \overline{\mathcal{F}}$ 
15 do  $\overline{M}_{i,j} \leftarrow 1$ 
16     $i \leftarrow i + 1$ 
17  $M^T \leftarrow (\widehat{M}^T, \widetilde{M}^T, \overline{M}^T)^T$ 
18 COMBINE( $M$ )
19 for  $i = 0$  to  $|\mathcal{F}| - 1$ 
20 do  $Z_i \leftarrow \{j | M_{i,j} = 1\}$ 
21 return ( $\mathcal{F}, Z$ )

```

Рис. 2.5. Построение ограничений для замороженных символов

цу. Строки матрицы  $H$  должны принадлежать пространству строк проверочной матрицы  $(2^m, k)$  итогового полярного кода, поскольку символы кодового слова из множества  $\widehat{\mathcal{F}}$  должны быть равны 0. В 8-ой строке задаваемые матрицей  $H$  ограничения на символы кодового слова преобразуются в задаваемые матрицей  $\widehat{M}$  ограничения на входные символы поляризующего преобразования. Строка 10 соответствует преобразованию ограничений  $\widetilde{Z}$  (для символов из  $\widetilde{\mathcal{F}}$ ) в матрицу  $\widetilde{M}$ . Функция  $Combine(M)$  на строке 18 выполняет операции над строками матрицы  $M$  с целью получения единичной матрицы из столбцов с индексами из множества  $\mathcal{F}$ .

Посредством выбора множества замороженных символов  $\widetilde{\mathcal{F}}$  и соответствующих ограничений  $\widetilde{Z}$  при построении укороченного полярного кода можно обеспечить достаточно большое минимальное расстояние  $d$ . Ограничения  $\widetilde{Z}$  могут быть получены из проверочной матрицы некоторого  $(N = 2^m, K \geq k + N - n, d)$  линейного кода, как описано в разделе 2.1.

#### 2.2.4. Оценка снизу для вероятности ошибки декодирования

Метрики вершин дерева укорочения являются оценками снизу для вероятности ошибки декодирования методом последовательного исключения  $(n = 2^m - s, k)$  кода, полученного путем укорочения  $(2^m, k + s)$  полярного кода с произвольным множеством замороженных символов  $\mathcal{F}$  при использовании произвольного шаблона  $S_m$ ,  $\text{wt}(S_m) = s$ . Предлагаемый подход для вычисления такой метрики использует таблицу вычисленных заранее оценок снизу для вероятностей ошибки  $P_{\lambda, i}^{(h)}$  в  $i$ -ых битовых подканалах  $2^{m-\lambda} \times 2^{m-\lambda}$  поляризующего преобразования, при условии того, что  $h \leq 2^{m-\lambda}$  символов передаются по абсолютно надежному каналу. Заметим, что  $P_{m, 0}^{(0)}$  равна вероятности ошибки в исходном канале передачи данных  $P(y|c)$ , и  $P_{m, 0}^{(1)} = 0$ . Можно показать, что  $\Psi^{(b)}(p_0, p_1)$ ,  $b \in \{0, 1\}$ , является возрастающей функцией от  $p_0$  и  $p_1$ . Это позволяет присвоить  $P_{\lambda, i+b\eta}^{(h)}$  наименьшие значения среди вычисленных согласно выражению (1.18), то

есть

$$P_{\lambda, i+b\eta}^{(h)} = \min_{j \in J} \Psi^{(b)}(P_{\lambda+1, i}^{(j)}, P_{\lambda+1, i}^{(h-j)}), \quad b \in \{0, 1\}, \quad (2.7)$$

где  $0 \leq \lambda < m$ ,  $\eta = 2^{m-\lambda-1}$ ,  $0 \leq i < \eta$  и  $J = \{\max(0, h - \eta), \dots, \lfloor h/2 \rfloor\}$ .

Оценка снизу для вероятности ошибки декодирования методом последовательного исключения для  $(n, k)$  укороченного полярного кода может быть получена как

$$\pi(m, n, k) = \min_{|\mathcal{F}|=2^m-k} \left( 1 - \prod_{\substack{i=0 \\ i \notin \mathcal{F}}}^{2^m-1} (1 - P_{0, i}^{(2^m-n)}) \right).$$

Заметим, что  $\Psi^{(0)}(p_0, p_1) = 0$  только если  $p_0 = 0$  и  $p_1 = 0$ , и  $\Psi^{(1)}(p_0, p_1) = 0$  если хотя бы  $p_0 = 0$  или  $p_1 = 0$ . Следовательно, среди  $P_{\lambda, 0}^{(h)}, P_{\lambda, 1}^{(h)}, \dots, P_{\lambda, 2^{m-\lambda}-1}^{(h)}$  должно быть не менее  $h$  нулевых значений. Однако для любого фиксированного шаблона веса  $h$  применение (1.18) приводит к появлению равно  $h$  нулевых значений. Это позволяет построить уточненную оценку снизу для вероятности ошибки декодирования методом последовательного исключения укороченного полярного кода. Пусть  $\tilde{P}_{\lambda, i}^{(h)}$  — оценка снизу ненулевой вероятности ошибки в  $i$ -ом битовом подканале  $2^{m-\lambda} \times 2^{m-\lambda}$  поляризирующего преобразования, которая равна

$$\tilde{P}_{\lambda, i}^{(h)} = \begin{cases} P_{\lambda, i}^{(h)}, & P_{\lambda, i}^{(h)} > 0, \\ \min_{j \in J_{\lambda+1, i}^{(0)}} \tilde{P}_{\lambda+1, i}^{(j)}, & P_{\lambda, i}^{(h)} = 0, i \in Y_{\lambda+1}^{(0)}, \\ \min_{j \in J_{\lambda+1, i}^{(1)}} \Psi^{(1)}(\tilde{P}_{\lambda+1, i}^{(j)}, \tilde{P}_{\lambda+1, i}^{(h-j)}), & P_{\lambda, i}^{(h)} = 0, i \in Y_{\lambda+1}^{(1)}, \\ 0, & P_{\lambda, i}^{(h)} = 0, i \notin Y_{\lambda+1}^{(0)} \cup Y_{\lambda+1}^{(1)}, \end{cases} \quad (2.8)$$

где  $J_{\lambda+1, i}^{(0)} = \{j' | \tilde{P}_{\lambda+1, i}^{(j')} > 0, j' \in \{j, h-j\}, j \in J\}$ ,  $J_{\lambda+1, i}^{(1)} = \{j \in J | (\tilde{P}_{\lambda+1, i}^{(j)} > 0) \wedge (\tilde{P}_{\lambda+1, i}^{(h-j)} > 0)\}$ ,  $Y_{\lambda+1}^{(0)} = \{i | (0 \leq i < 2^{m-\lambda-1}) \wedge (J_{\lambda+1, i}^{(0)} \neq \emptyset)\}$  и  $Y_{\lambda+1}^{(1)} = \{i | (2^{m-\lambda-1} \leq i < 2^{m-\lambda}) \wedge (J_{\lambda+1, i}^{(1)} \neq \emptyset)\}$ . Второй случай в (2.8) следует из того, что если  $p_b = 0$ , то  $\Psi^{(0)}(p_0, p_1) = p_{1-b}$ ,  $b \in \{0, 1\}$ .

Напомним, что применение любого шаблона для укорочения  $S_m$  веса  $s$  приводит к появлению множества динамически замороженных символов  $\hat{\mathcal{F}}$  такого,

что для любого  $i \in \widehat{\mathcal{F}}$  вероятность ошибки в битовом подканале  $p_{0,i}$  равна 0. Некоторые из этих подканалов являются общими для всех шаблонов веса  $s$ , множество таких подканалов равно  $\widetilde{V} = \{i | \widetilde{P}_{0,i}^{(s)} = 0\}$ . Битовые подканалы, которые могут стать динамически замороженными хотя бы для одного шаблона, составляют множество  $V = \{i | P_{0,i}^{(s)} = 0\}$ . Таким образом,  $\widetilde{V} \subset \widehat{\mathcal{F}} \subset V$ . При наличии предопределенного множества замороженных символов  $\widetilde{\mathcal{F}}$  должно выполняться  $\widetilde{\mathcal{F}} \subset \mathcal{F}$ . Остальные замороженные символы  $\mathcal{F} \setminus (\widehat{\mathcal{F}} \cup \widetilde{\mathcal{F}})$ , необходимые для задания  $(n, k)$  укороченного полярного кода, могут быть определены как  $n - k$  символов с наибольшими вероятностями  $p_{0,i} \geq \widetilde{P}_{0,i}^{(s)}$ ,  $i \in \{0, \dots, 2^m - 1\} \setminus (\widehat{\mathcal{F}} \cup \widetilde{\mathcal{F}})$ . Представленная на Рис. 2.9 функция  $FrozenSymbols(n, k, \widehat{\mathcal{F}}, \widehat{S})$  реализует эти вычисления.

Для получения оценки снизу для вероятности ошибки декодирования методом последовательного исключения достаточно построить  $\mathcal{F} \setminus \widetilde{V}$  как множество из  $2^m - k - |\widetilde{V}|$  индексов  $i \in \{0, \dots, 2^m - 1\} \setminus \widetilde{V}$  с наибольшими  $\widetilde{P}_{0,i}^{(s)}$ , при условиях  $|\mathcal{F} \cap V| \geq s$  и  $\widetilde{\mathcal{F}} \subset \mathcal{F}$ . Таким образом, получаем следующую уточненную оценку снизу

$$\pi'(m, n, k) = 1 - \prod_{\substack{i=0 \\ i \notin \mathcal{F}}}^{2^m-1} (1 - \widetilde{P}_{0,i}^{(2^m-n)}). \quad (2.9)$$

Для множества шаблонов, задаваемых массивом  $\widehat{S}$ , можно построить более точную оценку снизу для вероятности ошибки декодирования. Для заданных  $l$  и  $\phi$ , указывающих позицию последнего проинициализированного символа  $\widehat{S}_{l+1, 2\phi+1}$ , рассмотрим множество  $\mathcal{G}_r$  шаблонов  $S_m$ , соответствующих массиву  $\widehat{S}$ . Из выражения (2.4) следует, что такие шаблоны должны удовлетворять следующим условиям:

$$\text{wt}(S_{m, j2^{m-l-1}}, \dots, S_{m, (j+1)2^{m-l-1}-1}) = \widehat{S}_{l+1, j} \quad (2.10)$$

для  $0 \leq j \leq 2\phi + 1$  и

$$\text{wt}(S_{m, j2^{m-l}}, \dots, S_{m, (j+1)2^{m-l}-1}) = \widehat{S}_{l, j} \quad (2.11)$$

```

SUBCHANNELLB( $n, k, \lambda, \phi, \tilde{\mathcal{F}}, \hat{S}$ )
1   $m \leftarrow \lceil \log(n) \rceil$ 
2  for  $i = 0, \dots, 2\phi + 1$ 
3  do  $h \leftarrow \hat{S}_{\lambda+1, i}$ 
4    for  $j = 0$  to  $2^{m-\lambda-1} - 1$ 
5      do  $R_{\lambda+1, i2^{m-\lambda-1}+j} \leftarrow P_{\lambda+1, j}^{(h)}$ 
6         $\tilde{R}_{\lambda+1, i2^{m-\lambda-1}+j} \leftarrow \tilde{P}_{\lambda+1, j}^{(h)}$ 
7  for  $i = \phi + 1$  to  $2^\lambda - 1$ 
8  do  $h \leftarrow \hat{S}_{\lambda, i}$ 
9    for  $j = 0$  to  $2^{m-\lambda} - 1$ 
10   do  $R_{\lambda, i2^{m-\lambda}+j} \leftarrow P_{\lambda, j}^{(h)}$ 
11      $\tilde{R}_{\lambda, i2^{m-\lambda}+j} \leftarrow \tilde{P}_{\lambda, j}^{(h)}$ 
12 for  $i = 0$  to  $\phi$ 
13 do INITBLOCK( $m, \lambda, i, R$ )
14   INITBLOCK( $m, \lambda, i, \tilde{R}$ )
15 for  $\lambda' = \lambda - 1$  to  $0$ 
16 do for  $i = 0$  to  $2^{\lambda'} - 1$ 
17   do INITBLOCK( $m, \lambda', i, R$ )
18     INITBLOCK( $m, \lambda', i, \tilde{R}$ )
19 return ( $R, \tilde{R}$ )

```

Рис. 2.6. Вычисление оценок снизу для вероятностей ошибки в битовых подканалах

```

INITBLOCK( $m, \lambda, i, \Gamma$ )
1   $t \leftarrow 2^{m-\lambda-1}$ 
2  for  $j = 0, \dots, t - 1$ 
3  do for  $b \in \{0, 1\}$ 
4    do  $\Gamma_{\lambda, 2it+bt+j} \leftarrow \Psi^{(b)}(\Gamma_{\lambda+1, 2it+j}, \Gamma_{\lambda+1, 2it+t+j})$ 

```

Рис. 2.7. Вычисление вероятностей ошибки в битовых подканалах

```

LOWERBOUND( $n, k, l, \phi, \tilde{\mathcal{F}}, \hat{S}$ )
1  ( $R, \tilde{R}$ )  $\leftarrow$  SUBCHANNELLB( $n, k, l, \phi, \tilde{\mathcal{F}}, \hat{S}$ )
2   $m \leftarrow \lceil \log(n) \rceil$ 
3   $p \leftarrow 1$ 
4   $V = \{j \in \{0, \dots, 2^m - 1\} \mid R_{0,j} = 0\}$ 
5   $\tilde{V} = \{j \in \{0, \dots, 2^m - 1\} \mid \tilde{R}_{0,j} = 0\}$ 
6  if  $\tilde{\mathcal{F}} \cap \tilde{V} \neq \emptyset$ 
7    then return null
8   $\Lambda \leftarrow \{0, \dots, 2^m - 1\} \setminus (\tilde{\mathcal{F}} \cup \tilde{V})$ 
9  for  $i = 0$  to  $k - 1$ 
10 do if  $|V \cap \Lambda| + |\tilde{V}| = \hat{S}_{0,0}$ 
11   then  $\Lambda \leftarrow \Lambda \setminus V$ 
12    $\alpha \leftarrow \arg \min_{j \in \Lambda} \tilde{R}_{0,j}$ 
13    $\Lambda \leftarrow \Lambda \setminus \{\alpha\}$ 
14    $p \leftarrow p(1 - \tilde{R}_{0,\alpha})$ 
15  $p \leftarrow 1 - p$ 
16 return  $p$ 

```

Рис. 2.8. Вычисление метрики

```

FROZENSYMBOLS( $n, k, \tilde{\mathcal{F}}, \hat{S}$ )
1   $m \leftarrow \lceil \log(n) \rceil$ 
2  ( $R, \tilde{R}$ )  $\leftarrow$  SUBCHANNELLB( $n, k, m - 1, 2^{m-1} - 1, \tilde{\mathcal{F}}, \hat{S}$ )
3   $\mathcal{U} \leftarrow \{0, \dots, 2^m - 1\}$ 
4   $\hat{\mathcal{F}} \leftarrow \{j \in \mathcal{U} \mid R_{0,j} = 0\}$ 
5   $\mathcal{F} = \tilde{\mathcal{F}} \cup \hat{\mathcal{F}}$ 
6  for  $i = 0$  to  $n - k - |\tilde{\mathcal{F}}| - 1$ 
7  do  $\alpha \leftarrow \arg \max_{j \in \mathcal{U} \setminus \mathcal{F}} R_{0,j}$ 
8     $\mathcal{F} \leftarrow \mathcal{F} \cup \{\alpha\}$ 
9  return ( $\mathcal{F}, \hat{\mathcal{F}}$ )

```

Рис. 2.9. Построение множества замороженных символов

для  $\phi < j < 2^l$ .

Таким образом, для любых  $\lambda, i$  таких, что  $(\lambda < l+1) \vee ((\lambda = l+1) \wedge (i < 2\phi))$ , нет необходимости в выполнении непосредственной минимизации в выражениях (2.7) и (2.8), достаточно взять  $j = \widehat{S}_{\lambda,i}$ . Полученную таким способом оценку снизу для вероятности ошибки декодирования в  $i$ -ом промежуточном битовом подканале слоя  $\lambda$  обозначим  $R_{\lambda,i}, \widetilde{R}_{\lambda,i}$ . Функция *SubchannelLB*, приведенная на Рис. 2.6, реализует данный подход. Строки 2–11 соответствуют инициализации некоторых элементов массивов  $R$  и  $\widetilde{R}$  вычисленными заранее элементами массивов  $P$  и  $\widetilde{P}$ . Рекурсивное вычисление оценок вероятностей ошибки декодирования в битовых подканалах  $R_{0,i}$  и  $\widehat{R}_{0,i}$ ,  $0 \leq i < N$ , выполняется на строках 12–18, используемая при вычислениях функция *InitBlock* приведена на Рис. 2.7.

На Рис. 2.8 представлена предлагаемая функция *LowerBound*( $n, k, l, \phi, \widetilde{\mathcal{F}}, \widehat{S}$ ), предназначенная для вычисления оценки снизу для вероятностей ошибки декодирования методом последовательного исключения  $(n, k)$  кодов, которые могут быть получены путем укорочения полярного кода длины  $N = 2^m$ , множество замороженных символов которого состоит из  $\widetilde{\mathcal{F}}$  и некоторого множества из  $N - k - |\widetilde{\mathcal{F}}|$  символов. Заметим, что символы, принадлежащие множеству  $\widetilde{V}$  или  $\widetilde{\mathcal{F}}$ , являются замороженными для любого такого кода. Кроме того, не менее  $s - |\widetilde{V}|$  замороженных символов должно принадлежать  $V \setminus \widetilde{V}$ , где  $s = N - n = \widehat{S}_{0,0}$ . При вычислениях выражение (2.9) используется с  $\widetilde{R}$  вместо  $\widetilde{P}^{(2^m-n)}$ . На каждой итерации цикла *FOR* из множества  $\Lambda$  выбирается информационный символ с наименьшей ненулевой оценкой вероятности ошибки, и соответствующий сомножитель включается в (2.9).

### 2.2.5. Снижение сложности оптимизации

Заметим, что использование вышеописанного оптимизационного алгоритма с недостаточно большим максимальным размером приоритетной очереди  $q$



может привести к потере некоторых поддеревьев дерева укорочения (см. строку 18 алгоритма *JointOptimization*), и получению субоптимального результата. Однако сложность предлагаемого оптимизационного алгоритма при  $q = \infty$  становится чрезмерно высокой при  $n > 64$ .

Представим число укороченных символов  $s = 2^m - n$ , где  $m = \lceil \log_2(n) \rceil$ , как

$$s = \sum_{i=0}^{\mu} a_i 2^i,$$

где  $a_i \in \{0, 1\}$  для  $0 \leq i < \mu$  и  $a_\mu \geq 1$ ,  $\mu < m$ . Мы предлагаем субоптимальный алгоритм укорочения для длинных полярных кодов, осуществляющий поиск шаблонов  $S_m$  веса  $s$ , все элементы которых равны нулю за исключением непересекающихся блоков  $\mathcal{B}_{i,j} = \{f_{i,j}2^i, f_{i,j}2^i + 1, \dots, f_{i,j}2^i + 2^i - 1\}$  размера  $2^i$ ,  $0 \leq i \leq \mu$ ,  $0 \leq j < a_i$ .

Предлагаемый подход к снижению сложности основан на практических результатах, полученных для коротких полярных кодов ( $n \leq 64$ ), которые свидетельствуют о том, что оптимальный шаблон укорочения (вычисляемый *JointOptimization*) обычно состоит из блоков такого вида.

Применение данного ограничения для рассматриваемых шаблонов ведет к незначительным изменениям в *JointOptimization*. Перечислим данные изменения. Необходимо добавить входной аргумент  $a = (a_0, \dots, a_\mu)$ . Пути, хранимые в приоритетной очереди  $Q$ , из кортежей  $(p, l, \phi, \widehat{S})$  преобразуются в кортежи  $(p, l, \phi, \widehat{S}, \widehat{a})$ , где  $\widehat{a}_i$  — число свободных (еще неиспользованных) блоков  $2^i$ . Первоначально  $\widehat{a} = a$ . Для заданного пути  $(p, l, \phi, \widehat{S}, \widehat{a})$  следует пропустить итерации цикла *FOR*, при которых  $\widehat{a}_{\widehat{S}_{l+1,2\phi}} = 0$  или  $\widehat{a}_{\widehat{S}_{l+1,2\phi+1}} = 0$ . Прочие итерации выполняются согласно описанию на Рис. 2.4, за исключением того, что

- если  $\phi = 2^l - 1$ , то  $Push(Q, (p', l + 1, 0, \widehat{S}, \widehat{a} = a))$ ;
- в противном случае, уменьшить  $\widehat{a}_{\widehat{S}_{l+1,2\phi}}$  и  $\widehat{a}_{\widehat{S}_{l+1,2\phi+1}}$  на 1, и  $Push(Q, (p', l, \phi + 1, \widehat{S}, \widehat{a}))$ .

Заметим, что при  $\mu = 0$  получается первоначальный оптимизационный алгоритм, представленный на Рис. 2.4, поскольку при таком  $\mu$  есть  $s = 2^m - n$  блоков размера 1.

Для заданного  $a$  можно получить улучшенные оценки  $P$  и  $\tilde{P}$ , используемые в функции *SubchannelLB*, ведущие к построению более точной оценки для вероятности ошибки декодирования. Рассмотрим случай  $n$  кратного  $2^\mu$ , при этом шаблон укорочения состоит из  $a_\mu$  блоков размера  $2^\mu$ . Таким образом, каждое значение  $\hat{S}_{\lambda,i}$ ,  $0 \leq \lambda \leq m - \mu$ , кратно  $2^\mu$ , следовательно, в выражениях (2.7), (2.8) достаточно рассматривать только кратные  $2^\mu$  значения  $j$  и  $h - j$ ,  $0 \leq \lambda \leq m - \mu$ .

### 2.2.6. Сложность декодирования укороченных кодов

Алгоритм последовательного исключения, а также его списочные и стековые версии [19, 72], требует вычисления метрик, задаваемых выражением 1.11, для всех промежуточных символов поляризующего преобразования. При переиспользовании результатов промежуточных вычислений сложность вычисления всех таких значений составляет  $O(N \log(N))$  элементарных операций. Если алгоритм последовательного исключения оперирует ЛОПП, вычисляемыми согласно (1.14)–(1.15), то под элементарной операцией понимается применение любого из этих выражений.

Декодирование укороченного полярного кода может быть реализовано посредством любого алгоритма декодирования для соответствующего неукороченного кода при условии того, что символы кодового слова  $c_i = 0, i \in \text{supp}(S_m)$ , считаются абсолютно надежными, то есть  $L_{m,i} = \infty$ . В результате чего,  $s = \text{wt}(S_m)$  символов на каждом слое поляризующего преобразования считаются абсолютно надежными. Заметим, что если в выражениях (1.14)–(1.15) любое из значений  $L_{\lambda+1,i}, L_{\lambda+1,i+\eta}$  равно  $\pm\infty$ , то эти выражения становятся тривиальными и могут быть исключены из рассмотрения при подсчете сложности.

Пары вида  $(L_{\lambda,i}, L_{\lambda,i+\eta})$  будем называть парами типа  $T_\alpha$ , если  $\alpha$  ее элемен-

тов равны  $\pm\infty$ . Пусть  $\tau_\alpha$  — общее число элементов, входящих в пары типа  $T_\alpha$ , тогда число элементарных операций можно выразить как  $N \log(N) - \tau_1 - \tau_2$ . Для того, чтобы получить верхнюю границу для этой величины, заметим что  $\tau_1/2 + \tau_2 = s \log(N)$ , поскольку общее количество промежуточных символов с ЛОПП  $\pm\infty$  равно  $s \log(N)$  и  $\alpha/2$ ,  $\alpha = 1, 2$ , элементов в парах типа  $T_\alpha$  имеет ЛОПП  $\pm\infty$ . Кроме того, структура поляризирующего преобразования предполагает, что  $\tau_2 \leq s \lceil \log(s) \rceil$ . Таким образом,  $\tau_1 + \tau_2 = 2s \log(N) - \tau_2 \geq 2s \log(N) - s \lceil \log(s) \rceil = s(\log(N) + \lceil \log(N/s) \rceil)$ . Следовательно, число элементарных операций не превосходит  $N \log(N) - s(\log(N) + \lceil \log(N/s) \rceil)$ .

Сложность алгоритма последовательного исключения при декодировании полярного кода длины  $N$ , укороченного на  $s$  символов, составляет  $O((N - s) \log(N) - s \lceil \log(N/s) \rceil)$ . В случае списочного алгоритма последовательного исключения приведенную оценку сложности следует умножить на размер списка  $L$ .

### 2.2.7. Численные результаты

Таблица 2.1 характеризует сложность предлагаемого алгоритма совместной оптимизации для укороченных полярных кодов скорости 0.5, построенных для  $N = 64$ ,  $E_b/N_0 = 5$  дБ и  $q = \infty$ . Столбцы  $n$ ,  $\binom{N}{s}$  и  $\mathcal{M}^{(N)}(s)$  соответствуют длине укороченного кода, числу существующих шаблонов укорочения веса  $s = N - n$  и числу неэквивалентных шаблонов укорочения, соответственно. Остальные столбцы содержат наибольшее число путей, хранимых в приоритетной очереди  $Q$  во время выполнения оптимизации (потребление памяти), число итераций цикла *WHILE* в алгоритме *JointOptimization*, которое равно числу посещенных вершин в дереве укорочения, и вероятность ошибки декодирования методом последовательного исключения  $P^{(e)}$ , вычисленную согласно выражению (1.12) для построенного укороченного полярного кода. Параметр  $\mu$  задает размер блоков в шаблоне укорочения (см. раздел 2.2.5). Видно, что число посе-

Таблица 2.1. Сложность предлагаемого алгоритма совместной оптимизации при  $N = 64$

$N - s$	$\binom{N}{s}$	$\mathcal{M}^{(N)}(s)$	Число шаблонов			Итерации <i>WHILE</i>			$P^{(e)}$	
			$\mu = 0$	$\mu = 1$	$\mu = 2$	$\mu = 0$	$\mu = 1$	$\mu = 2$	$\mu = 0$	$\mu = 2$
34	1.62E18	2.37E7	76229	1131	113	79220	2751	601	0.00165	0.00174
36	1.11E18	1.97E7	66599	1185	28	67760	3944	175	0.00173	0.00236
38	6.01E17	1.44E7	76363	1020	97	94417	4314	607	0.00196	0.00270
40	2.50E17	9.37E6	22699	601	27	22481	1715	157	0.00124	0.00124
42	8.03E16	5.35E6	13012	419	70	12312	1092	317	0.00097	0.00097
44	1.96E16	2.68E6	8050	388	17	6095	920	183	0.00071	0.00071
46	3.60E15	1.17E6	14974	430	48	14265	1045	431	0.00074	0.00074
48	4.88E14	4.52E5	10752	325	14	12311	962	154	0.00065	0.00065
50	4.78E13	1.50E5	8357	204	26	11710	964	321	0.00071	0.00071
52	3.28E12	4.31E4	4629	121	6	9573	941	79	0.00069	0.00069
54	1.51E11	1.05E4	4808	55	12	19100	497	186	0.00067	0.00075
56	4.42E9	2.20E3	1425	30	4	7450	380	58	0.00062	0.00071
58	7.49E7	3.81E2	270	10	4	1660	158	84	0.00060	0.00074
60	6.35E5	5.5E1	44	4	1	302	90	32	0.00053	0.00066
62	2.01E3	6	6	1	1	62	32	32	0.00047	0.00049
64	1	1	1	1	1	32	32	32	0.00044	0.00044

ценных вершин в дереве укорочения даже при  $\mu = 0$  значительно меньше, чем  $\mathcal{M}^{(N)}(s)$ , равное общему числу листьев в дереве укорочения. Заметим, что объем потребляемой памяти и вычислительная сложность предлагаемого оптимизационного алгоритма убывают с ростом  $\mu$ . С другой стороны, увеличение  $\mu$  ведет к незначительному снижению корректирующей способности получаемых кодов. Заметим, что для очень больших  $s$  вероятность ошибки декодирования получаемых кодов возрастает с  $n$ , что является следствием как субоптимальности алгоритма последовательного исключения, так и того, что укороченные полярные коды длины  $2^{m-1} + t$  при малых  $t$  могут иметь меньшее минимальное расстояние, чем полярные коды длины  $2^{m-1}$  с аналогичной скоростью.

На Рис. 2.10 представлен график зависимости вероятности ошибки декоди-

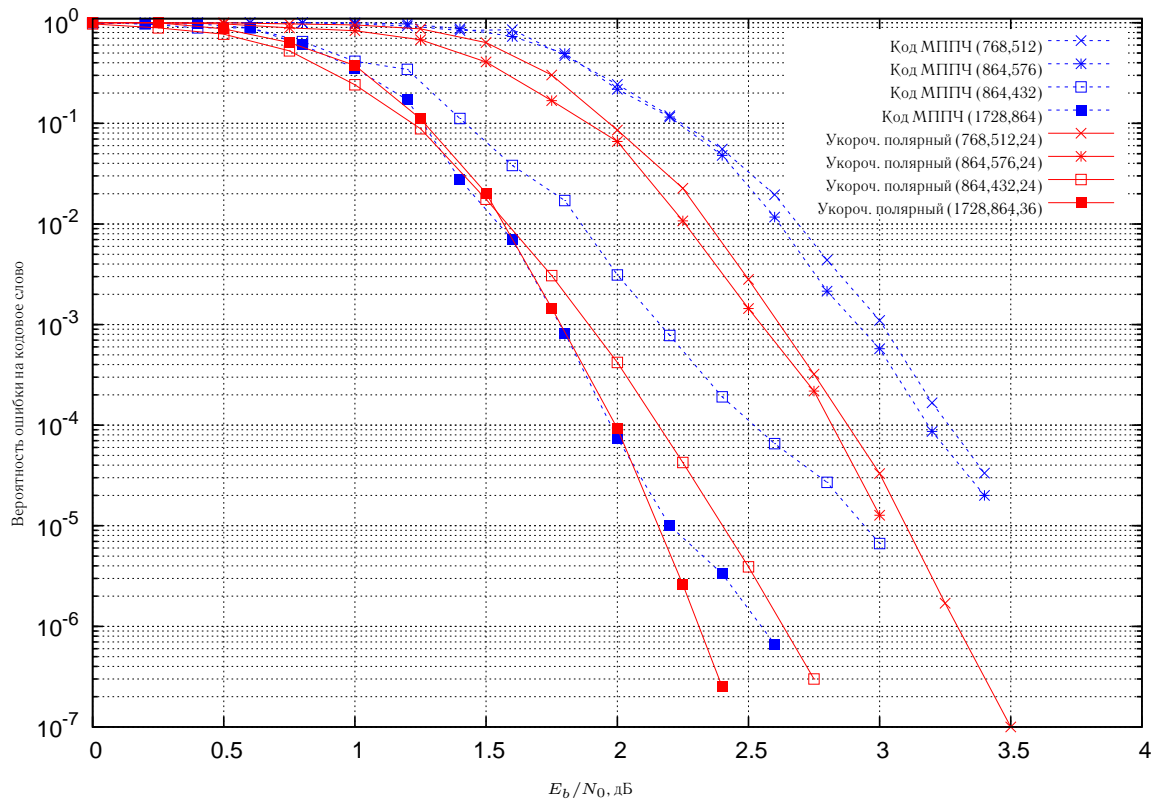


Рис. 2.10. Сравнение корректирующей способности укороченных полярных кодов и LDPC

рования от отношения сигнал/шум для укороченных полярных кодов, построенных с помощью предлагаемого метода. Данные коды построены с использованием  $\tilde{\mathcal{F}}, \tilde{\mathcal{Z}}$ , задаваемого расширенным кодом БЧХ. Декодирование этих кодов выполнялось с помощью стекового алгоритма последовательного исключения, описанного в разделе 3.2 с  $L = 128$ . Для сравнения показана корректирующая способность кодов МППЧ из стандарта WiMAX. Видно, что (768, 512, 24) и (864, 576, 24) укороченные полярные коды значительно превосходят соответствующие коды МППЧ. Однако (1728, 864, 36) укороченный полярный код обеспечивает некоторый выигрыш только при больших отношениях сигнал/шум, при которых корректирующая способность определяется в основном минимальным расстоянием.

### 2.2.8. Выводы

В данном разделе был предложен новый метод построения укороченных полярных кодов, в основе которого лежит совместная оптимизация шаблона укорочения и множества замороженных битовых подканалов с целью минимизации вероятности ошибки декодирования методом последовательного исключения. Предлагаемый подход позволяет обеспечить достаточно большое минимальное расстояние получаемого кода путем использования соответствующего алгебраического кода, к примеру, расширенного кода БЧХ, в качестве начальной точки для оптимизации. Декодирование укороченного полярного кода может быть реализовано с помощью алгоритма последовательного исключения и его аналогов.

## 2.3. Построение полярных кодов с произвольным двоичным ядром

Задача построения  $(n = l^m, k)$  классического двоичного полярного кода с  $l \times l$  ядром  $M$  сводится к задаче выбора  $k$  строк матрицы  $G_n = \Lambda^{(l,m)} M^{\otimes m}$ , соответствующих  $k$  лучшим битовым подканалам  $W_{G_n}^{(i)}$ . Вероятность ошибки в этих битовых подканалах зависит от распределений ЛОПП для символов  $u_i$ ,  $0 \leq i < n$ . Задача вычисления данных распределений в общем случае является крайне трудоемкой. Однако в случае двоичного стирающего канала каждый подканал  $\lambda$ -го слоя характеризуется вероятностью стирания, то есть вероятностью того, что декодер на  $\lambda$ -ом слое не способен восстановить соответствующий символ. Далее будет рассматриваться задача построения классических полярных кодов для случая двоичного стирающего канала, которая сводится к задаче оценки вероятности отказа от декодирования для информационных символов заданных линейных блочных кодов длины  $l$ .

Рассмотрим передачу кодового слова полярного кода по двоичному стирающему каналу с вероятностью стирания  $\bar{p}$ . Пусть  $P_\lambda[i]$  — вероятность стирания

для  $i$ -го символа  $\lambda$ -го слоя,  $0 \leq i < n$ ,  $0 \leq \lambda \leq m$ . На 0-ом слое, соответствующем принятой последовательности, вероятность стирания для всех символов равна  $P_0[i] = \bar{p}$ , поэтому на 1-ом слое будет всего  $l$  различных вероятностей  $P_\lambda[i]$ , соответствующих  $l$  строкам матрицы  $M$ . В общем случае, на слое  $\lambda$  содержится  $l^\lambda$  различных вероятностей  $P_\lambda[i]$ , при этом на вход компонентных декодеров поступают символы с одинаковыми вероятностями стирания. Таким образом, для слоя  $0 \leq \lambda < m$

$$P_{\lambda+1}[l\phi + \beta] = Q(P_\lambda[\phi], \beta), \quad 0 \leq \beta < l, 0 \leq \phi < l^\lambda, \quad (2.12)$$

где  $Q(p, \beta)$  — функция качества подканалов ядра, возвращающая вероятность отказа от декодирования для 0-го информационного символа кода, порожденного последними  $\mu = l - \beta$  строками матрицы  $M$ .

Если нам известна функция  $Q(p, \beta)$ , то мы можем рекурсивно вычислить  $P_\lambda[i]$ . Тогда построение порождающей матрицы для  $(n = l^m, k)$  классического полярного кода осуществляется путем выбора  $k$  строк матрицы  $G_n$ , соответствующих наименьшим значениям  $P_m[i]$ .

### 2.3.1. Вероятность отказа от декодирования информационного символа

Рассмотрим передачу кодовых слов вида  $u_0^{\mu-1}\Gamma$ , принадлежащих  $(l, \mu)$  коду с порождающей матрицей  $\Gamma$ , по двоичному стирающему каналу с вероятностью стирания  $p$ . Определим конфигурацию стираний как вектор  $E \in \{0, 1\}^l$ , где единицы соответствуют стертым позициям принятой последовательности  $y_0^{l-1}$ . Вычислим вероятность того, что при декодировании по максимуму правдоподобия невозможно восстановить значение символа  $u_0$ , то есть вероятность того, что конфигурация стираний является неисправимой. Конфигурация стираний называется неисправимой, если существуют как минимум два вектора  $u_0^{\mu-1}$  и  $v_0^{\mu-1}$  такие, что  $u_0 \neq v_0$  и  $(u_0^{\mu-1}\Gamma)_i = (v_0^{\mu-1}\Gamma)_i = y_i$  во всех нестертых позициях  $i \in \{0, \dots, l-1\} \setminus \text{supp}(E)$ . Заметим, что может существовать способ восстановления  $u_0$ , даже если кодовое слово целиком невозможна при заданной

конфигурации стираний.

Вероятность отказа от декодирования информационного символа может быть представлена в следующем виде

$$Q(p, \beta) = \sum_{e=d}^l B_e p^e (1-p)^{l-e}, \quad (2.13)$$

где  $d$  – минимальное расстояние рассматриваемого  $(l, \mu)$  кода,  $p$  – вероятность стирания символа,  $B_e$  – число неисправимых конфигураций стираний веса  $e$ ,  $0 \leq e \leq l$ .

### 2.3.2. Точный метод

В данном разделе представлен метод для вычисления распределения числа неисправимых конфигураций стираний  $B_e$ . При принятой последовательности  $y_0^{l-1}$  с  $e$  стираниями декодирование сводится к решению системы линейных уравнений

$$u_0^{\mu-1} \tilde{\Gamma} = \tilde{y}_0^{\eta-1}, \quad (2.14)$$

где  $\eta = l - e$ , последовательность  $\tilde{y}_0^{\eta-1}$  состоит из нестертых символов последовательности  $y_0^{l-1}$ , и  $\mu \times \eta$  матрица  $\tilde{\Gamma}$  состоит из столбцов матрицы  $\Gamma$ , соответствующих нестертым символам. Для вычисления значения  $u_0$  выберем вектор  $\tilde{z}_0^{\eta-1} \in \{0, 1\}^\eta$  такой, что

$$\tilde{\Gamma} \left( \tilde{z}_0^{\eta-1} \right)^T = \left( 1 \ 0 \ \dots \ 0 \right)^T. \quad (2.15)$$

Таким образом,  $u_0 = \tilde{y}_0^{\eta-1} \left( \tilde{z}_0^{\eta-1} \right)^T$ . Если вектор  $\left( 1 \ 0 \ \dots \ 0 \right)^T$  не принадлежит пространству столбцов матрицы  $\tilde{\Gamma}$ , то значение  $u_0$  не может быть однозначно найдено из уравнения (2.14).

Множество решений системы линейных уравнений (2.15) является подмножеством решений системы

$$\Gamma \left( z_0^{l-1} \right)^T = \left( 1 \ 0 \ \dots \ 0 \right)^T, \quad (2.16)$$



для элементов которого выполняется  $\text{supp}(z_0^{l-1}) \cap E = \emptyset$ . Уравнение (2.16) определяет все пути восстановления  $u_0$  из символов кодового слова, следовательно каждый вектор  $z$ , удовлетворяющий (2.16), задает выражение  $u_0 = \sum_{i:z_i=1} y_i$ , которое может быть использовано декодером, если не стерт ни один элемент  $y_i$ , участвующий в выражении. Рассмотрим множество

$$\bar{Y} = \left\{ E \mid E_i = 1 - z_i, \Gamma (z_0^{l-1})^T = \begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix}^T \right\}. \quad (2.17)$$

Все конфигурации стираний в  $\bar{Y}$  являются исправимыми. Заметим, что если некоторая конфигурация стираний  $E$  исправима, то все конфигурации  $E' : \text{supp}(E') \subset \text{supp}(E)$  также исправимы. Таким образом, функция

$$f(E) = \bigvee_{S \in \bar{Y}} \left( \bigwedge_{i:S_i=0} \neg E_i \right)$$

принимает значение “истина” тогда и только тогда, когда конфигурация стираний  $E$  является исправимой, в свою очередь функция  $\bar{f}(E)$ , являющаяся характеристической функцией  $\bar{Y}$ , задает базис для множества исправимых конфигураций стираний, то есть если  $\bar{f}(E) = 1$ , то  $f(E) = 1$ .

Для подсчета числа неисправимых конфигураций стираний  $B_e$  построим упорядоченное двоичное дерево решений (ДДР), задающее функцию  $f(E)$  [11, 17, 46]. Данное ДДР содержит  $l$  уровней. Ребра на  $s$ -ом уровне соответствуют возможным значениям  $E_s$ . Пути, начинающиеся в корне ДДР и ведущие в терминальные вершины “1” и “0”, задают множество исправимых и неисправимых конфигураций стираний, соответственно. Значение  $B_e$  равно количеству путей из корня дерева в терминальную вершину “0”, содержащих  $e$  переходов по нулю.

Из (2.17) видно, что  $\bar{Y}$  представляет собой смежный класс кода с проверочной матрицей  $\Gamma$ . По матрице  $\Gamma$  можно построить решетку для этого кода, которая преобразуется в ДДР, задающее функцию  $\bar{f}(E)$  принадлежности смежному классу  $\bar{Y}$  [45].

Опишем алгоритм преобразования ДДР для функции  $\bar{f}(E)$  в ДДР для функции  $f(E)$ . Предлагаемый алгоритм преобразования основан на том, что если

```

PROCESSBDD( $\overline{BDD}$ )
1   $BDD \leftarrow \overline{BDD}$ 
2  for  $s \leftarrow 0$  to  $l - 1$ 
3  do for  $j \leftarrow 1$  to  $|BDD_{s,-}|$ 
4      do  $BDD_{s,j}[0] \leftarrow BDD_{s,j}[0] \vee BDD_{s,j}[1]$ 
5  return  $BDD$ 

```

Рис. 2.11. Преобразование ДДР функции  $\overline{f}(E)$  в ДДР функции  $f(E)$

некоторая конфигурация стираний  $E$  является исправимой, то исправимой является и конфигурация  $E' : \text{supp}(E') \subset \text{supp}(E)$ . Пусть  $BDD_{s,j}[c]$ ,  $c \in \{0, 1\}$ , — поддерево ДДР, соответствующего функции  $\overline{f}(E)$ , которое задает некоторое множество суффиксов конфигураций стираний  $(E_s, \dots, E_{l-1})$  с  $E_s = c$ . Если существуют пути в  $BDD_{s,j}[1]$ , ведущие из корня в терминальную вершину “1”, то мы должны включить их в множество путей  $BDD_{s,j}[0]$ , ведущих из корня в терминальную вершину “1”, то есть нам необходимо заменить все поддеревья ДДР, соответствующего функции  $\overline{f}(E)$ , на  $s$ -ом уровне поддеревом, соответствующим как  $E_s = 1$ , так и  $E_s = 0$ . Предлагаемый метод обработки первоначального ДДР представлен на Рис. 2.11. Сложность этого алгоритма определяется профилем сложности решетки кода с проверочной матрицей  $\Gamma$ .

Предлагаемый подход для вычисления числа неисправимых конфигураций стираний  $V_e$  состоит из следующих этапов:

- Построение решетки, задающей смежный класс кода с проверочной матрицей  $\Gamma$ , определяемой (2.16).
- Преобразование решетки в ДДР  $\overline{BDD}$ , соответствующее функции  $\overline{f}(E)$ .
- Преобразование ДДР  $\overline{BDD}$  в ДДР  $BDD$ , соответствующее функции  $f(E)$ , с использованием алгоритма, представленного на Рис. 2.11.
- Вычисление  $V_e$  как количества путей в ДДР  $BDD$  из корня в терминальную вершину “0”.

### 2.3.3. Приближенный метод

При размерности поляризующего ядра  $M$  более 32 явное построение множества всех исправимых конфигураций стираний с помощью вышеприведенного алгоритма становится невозможным. В связи с этим возникает необходимость построения приближенного метода оценки вероятности отказа от декодирования первого информационного символа кода с порождающей матрицей  $\Gamma$ . Предлагаемый в данном разделе подход является обобщением метода [34].

**Теорема 1.** *При  $e < \lfloor (3d + 1)/2 \rfloor$  число неисправимых конфигураций стираний*

$$B_e = \sum_{i=d}^e \binom{l-i}{e-i} A_i, \quad (2.18)$$

где  $d$  — наименьший вес кодового слова  $c_0^{l-1} = u_0^{\mu-1} \Gamma$  при  $u_0 = 0$  и  $u_1^{\mu-1} \in \{0, 1\}^{\mu-1}$ ,  $A_i$  — число кодовых слов  $c_0^{l-1}$  веса  $i$ . При  $e \geq \lfloor (3d + 1)/2 \rfloor$  число неисправимых конфигураций стираний ограничено сверху

$$B_e \leq \min \left( \sum_{i=1}^e \binom{l-i}{e-i} A_i, \binom{l}{e} \right). \quad (2.19)$$

*Доказательство.* Напомним, что конфигурация стираний является неисправимой, если она покрывает хотя бы одно кодовое слово, соответствующее  $u_0 = 1$ . Для каждого кодового слова  $c_0^{l-1}$  веса  $i$  мы можем построить  $\binom{l-i}{e-i}$  конфигураций стираний веса  $e$ , которые бы покрывали все  $c_j = 1$ ,  $0 \leq j < l$ . Если  $e < \lfloor (3d + 1)/2 \rfloor$ , то множества таких конфигураций стираний, построенные для кодовых слов  $c_0^{l-1}$  и  $s_0^{l-1}$  не пересекаются. Действительно, кодовые слова различаются как минимум в  $t_1 \geq d$  позициях. Число позиций, на которых стоят единицы у обоих кодовых слов, равно  $t_2 \geq d - \lfloor t_1/2 \rfloor$ . Отсюда, если число стираний удовлетворяет условию  $e = t_1 + t_2 \geq t_1 + d - \lfloor t_1/2 \rfloor = d + \lfloor (t_1 + 1)/2 \rfloor \geq \lfloor (3d + 1)/2 \rfloor$ , то такие конфигурации стираний могут покрывать более одного кодового слова.

Если  $e \geq \lfloor (3d + 1)/2 \rfloor$ , то множества неисправимых конфигураций стираний для различных кодовых слов могут пересекаться. Таким образом, мы получаем

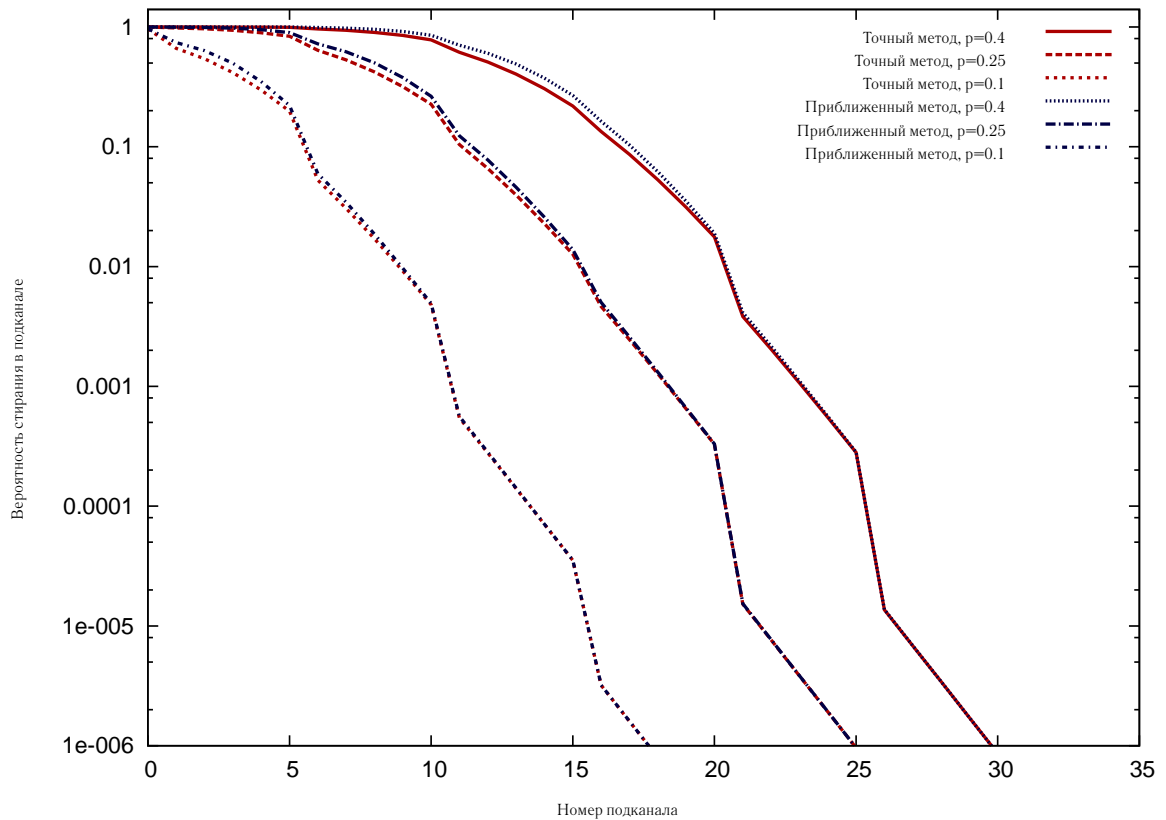


Рис. 2.12. Вероятность стирания в битовых подканалах  $32 \times 32$  ядра БЧХ

верхнюю границу  $B_e \leq \sum_{i=1}^e \binom{l-i}{e-i} A_i$ . С другой стороны, число конфигураций стираний не может превышать  $\binom{l}{e}$ .  $\square$

Заметим, что при оценке вероятности отказа от декодирования по формуле (2.13) нам наиболее важно знать точные значения  $B_e$  для небольших  $e$ . Поэтому использование верхней границы (2.19) при вычислении  $Q(p, \beta)$  не приводит к возникновению значительной погрешности.

### 2.3.4. Численные результаты

На Рис. 2.12 представлен график зависимости вероятности отказа от декодирования от номера битового подканала для  $32 \times 32$  ядра БЧХ, конструкция которого описана в разделе 1.1.2. При вычислении вероятностей использовался как точный метод, описанный в разделе 2.3.2, так и приближенный метод, описанный в разделе 2.3.3. Оптимизация кодов выполнялась для случая двоичного стирающего канала с вероятностью стирания  $p$ . Для подканалов с высокой ве-

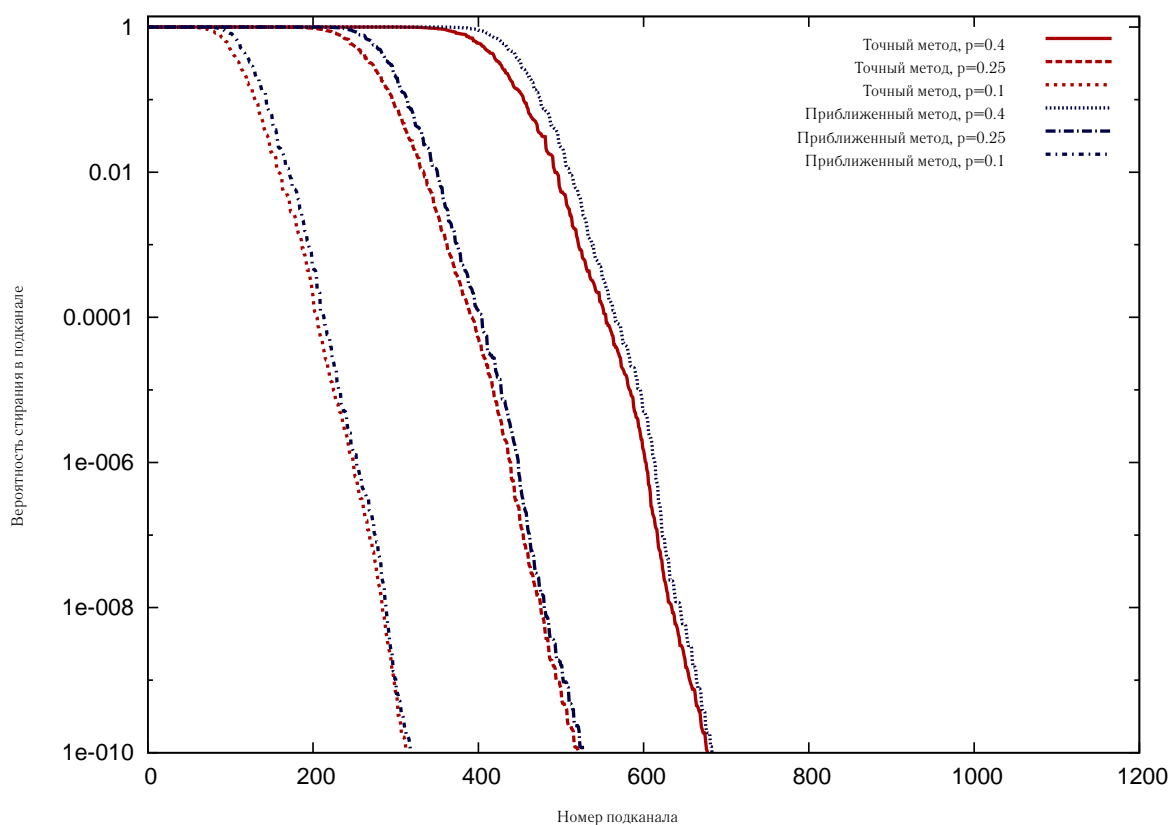


Рис. 2.13. Вероятность стирания в битовых подканалах полярного кода с  $32 \times 32$  ядром БЧХ

роятностью стирания приближенный метод дает менее точные оценки, чем для подканалов с малой вероятностью стирания. Рис. 2.13 характеризует качество битовых подканалов полярного кода  $(1024, 512)$ , построенного на базе  $32 \times 32$  ядра БЧХ. Заметим, что подканалы отсортированы по вероятностям стирания в них. Видно, что по мере снижения вероятности стирания в исходном канале передачи данных, увеличивается доля битовых подканалов с малыми вероятностями стирания. При рекурсивном вызове функции (2.12) происходит накопление погрешности.

На Рис. 2.14 представлен график зависимости вероятности ошибки декодирования методом последовательного исключения от отношения сигнал/шум для кода  $(1024, 512)$ . Код, построенный с помощью точного метода, демонстрирует лучшую корректирующую способность, чем код, построенный с помощью приближенного метода, однако разница незначительна. На Рис. 2.15 представлен аналогичный график для полярных кодов  $(4096, 2048)$ . Код, построенный на

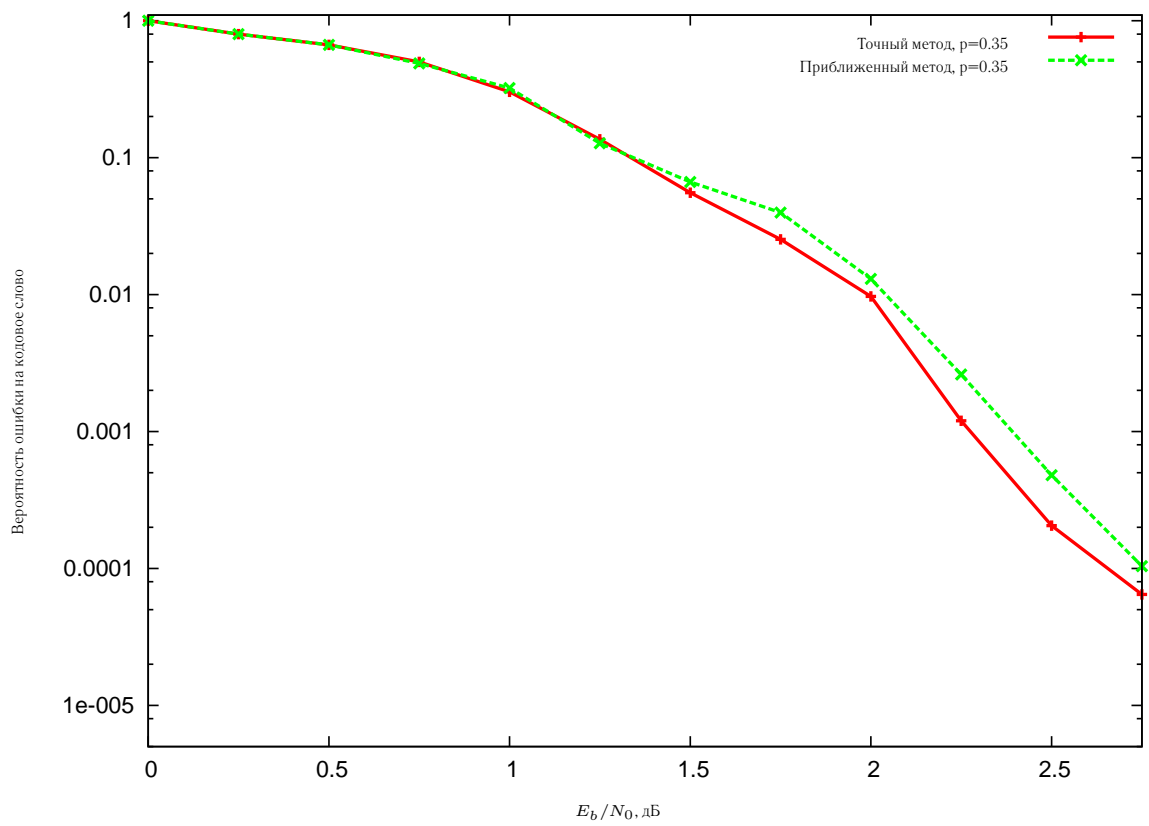


Рис. 2.14. Вероятность ошибки декодирования (1024, 512) полярного кода

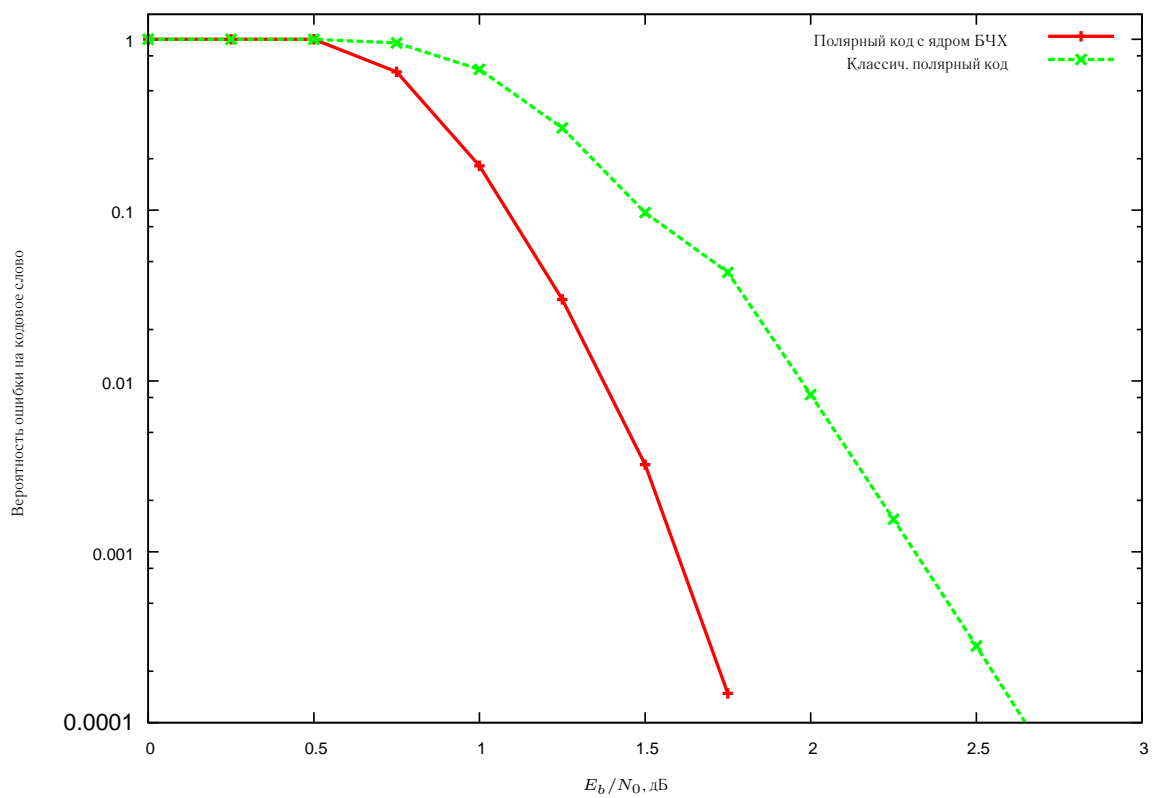


Рис. 2.15. Вероятность ошибки декодирования (4096, 2048) полярного кода

базе  $64 \times 64$  ядра БЧХ, обеспечивает энергетический выигрыш более 1 дБ по сравнению с кодом, построенным на базе  $2 \times 2$  ядра Арикана. Ввиду большой сложности точного метода в случае поляризующих ядер большой размерности, для построения кода (4096, 2048) на базе ядра  $64 \times 64$  использовался только приближенный метод.

### **2.3.5. Выводы**

В данном разделе был предложен алгоритм построения полярных кодов с произвольным двоичным ядром поляризации. Оптимизация полярного кода выполняется для случая двоичного стирающего канала и декодирования методом последовательного исключения. Выбор множества замороженных битовых подканалов осуществляется на основании вычисленных рекурсивно вероятностей отказа от декодирования для символов входной последовательности. Каждый шаг рекурсии выполняется на основании распределения числа неисправимых конфигураций стираний. Предложены два метода оценки числа неисправимых конфигураций стираний для кодов, порождаемых строками ядра поляризации. Первый метод является точным, он может быть использован для ядер размерности не более 32. Для ядер больших размерностей был предложен приближенный метод. Полученные численные результаты свидетельствуют о том, что построенные полярные коды демонстрируют высокую корректирующую способность и в случае Гауссовского канала.

## **2.4. Выводы по разделу**

Была предложена конструкция полярного подкода заданного расширенного кода БЧХ, обеспечивающая минимизацию вероятности ошибки декодирования методом последовательного исключения. Был предложен метод выбора полярных подкодов БЧХ заданной размерности, обеспечивающий минимизацию вероятности ошибки их декодирования заданным алгоритмом. Приведены примеры

$(n, k)$  полярных подкодов БЧХ, обеспечивающих меньшую вероятность ошибки декодирования при использовании списочного/стекового алгоритма последовательного исключения, чем  $(n, k)$  классический полярный код. Показано, что полярные подкоды БЧХ могут выигрывать по вероятности ошибки декодирования у кодов МППЧ и полярных кодов с CRC.

Предлагаемый алгоритм построения полярных кодов с произвольным двоичным ядром поляризации предполагает выбор множества замороженных битовых подканалов, оптимального для случая двоичного стирающего канала и декодирования методом последовательного исключения. Выбор этого множества осуществляется на основании вычисленных рекурсивно вероятностей отказа от декодирования для информационных символов. Предложены точный и приближенный методы для осуществления шагов рекурсии. Полученные численные результаты свидетельствуют о том, что построенные полярные коды демонстрируют высокую корректирующую способность и в случае Гауссовского канала.

Также был предложен новый метод построения укороченных полярных кодов с ядром Арикана, в основе которого лежит совместная оптимизация шаблона укорочения и множества замороженных битовых подканалов. Предлагаемый подход позволяет обеспечить достаточно большое минимальное расстояние получаемого кода путем использования соответствующего алгебраического кода, к примеру, расширенного кода БЧХ, в качестве начальной точки для оптимизации. Декодирование укороченного полярного кода может быть реализовано с помощью алгоритма последовательного исключения и его аналогов.



## Глава 3

# Декодирование полярных кодов

### 3.1. Декодирование методом направленного поиска

Декодирование полярного кода по максимуму апостериорной вероятности предполагает поиск последовательности  $u_0^{n-1}$ , при которой достигается максимум  $P(u_0^{n-1}|y_0^{n-1})$  с учетом наличия ограничений на значения замороженных символов.

#### 3.1.1. Описание алгоритма

Предлагаемый подход основан на стековом алгоритме последовательного исключения, описанном в разделе 1.3.3. Если на каждой итерации стекового алгоритма последовательного исключения выбирается путь  $u_0^{i-1}$ , соответствующий максимуму  $P(u_0^{i-1}|y_0^{n-1})$ , то декодер выполнит ровно  $n$  итераций. Однако для пути  $u_0^{i-1}$  декодер вычисляет вероятность  $P(u_0^{i-1}|y_0^{n-1})$ , что приводит к его частым переключениям между различными путями. В результате чего, число итераций, осуществляемых декодером при поиске наиболее вероятного кодового слова, оказывается существенно больше  $n$ . Для разрешения данной проблемы мы предлагаем представить вероятность  $P(u_0^{n-1}|y_0^{n-1})$  как  $P(u_0^{i-1}|y_0^{n-1}) \prod_{j=i}^{n-1} P(u_j|u_0^{j-1}, y_0^{n-1})$  и заменить сомножители  $P(u_j|u_0^{j-1}, y_0^{n-1})$  их математическими ожиданиями, вычисленным по всем возможным принятым последовательностям  $y_0^{n-1}$ . Заметим, что для правильного пути  $u_0^j$  такие математические ожидания совпадают с вероятностями принятия правильного решения в подканалах  $(1 - P_j)$ ,  $i \leq j < n$ . Таким образом, на каждой итерации предлагаемый декодер выбирает для удлинения путь  $u_0^{i-1}$ , которому соответствует наибольшая метрика  $P(u_0^{i-1}|y_0^{n-1})\psi(i)$ , где  $\psi(i) = \prod_{j=i}^{n-1} (1 - P_j)$ . Данный подход мож-

но рассматривать как случай  $A$ -алгоритма (см. [71] и ссылки) с эвристической функцией  $\psi(i)$ , характеризующей стоимость неисследованной части пути.

Пусть  $u_0^{n-1}$  соответствует наиболее вероятному кодовому слову. Если при декодировании для некоторого  $i$  окажется, что  $\psi(i) < \prod_{j=i}^{n-1} P(u_j|u_0^{j-1}, y_0^{n-1})$ , то произойдет ошибка декодирования в том случае, если существует кодовое слово  $c : P(c_0^{n-1}|y_0^{n-1}) > P(u_0^{i-1}|y_0^{n-1})\psi(i)$ . Результаты статистического моделирования показывают, что вероятность того, что правильный путь будет удален ввиду ограничений на размер стека (см. раздел 1.3.3) значительно превышает вероятность того, что неправильный путь будет расширен до фазы  $n$  по причине того, что реальная вероятность еще непостроенной части правильного пути превосходит  $\phi(i)$ .

Предлагаемый подход может быть эффективно реализован аналогично списочному алгоритму последовательного исключения (см. 1.3.4), как показано на Рис. 3.1. Алгоритм оперирует вероятностями, однако, он может быть легко преобразован для работы с логарифмами вероятностей путей. Инициализация структур данных выполняется так же, как в случае списочного алгоритма последовательного исключения. Пути в стеке (приоритетной очереди) упорядочены согласно значениям метрик  $M = P(u_0^{i-1}|y_0^{n-1})\psi(i)$ . На каждой итерации путь с наибольшей метрикой  $M$  извлекается из стека. Для этого пути с номером  $f$  вычисляются вероятности  $P_{f,m}[0][v]$ ,  $v \in \{0, 1\}$ . Если фаза  $\phi_f$  соответствует замороженному символу, то для расширения пути используется только значение  $FrozenValue(f, \phi_f)$ , задаваемое ограничением статической/динамической заморозки (2.1). В противном случае, путь копируется и рассматриваются оба варианта значения  $u_i \in \{0, 1\}$ . Величина  $q_i$  показывает, сколько раз пути длины  $i$  извлекались из стека. Чтобы предотвратить переполнение стека, если  $q_i \geq L$ , то все пути длины меньше  $i + 1$  удаляются из стека. Таким образом, число путей, находящихся на заданной фазе декодирования, ограничено параметром  $L$ . число путей в приоритетной очереди  $\tau$  превышает ее вместимость  $\Theta$ , то наиме-

```

DECODE( $y_0^{n-1}, L, \Theta$ )
1  INITIALIZEDATASTRUCTURES()
2   $f \leftarrow \text{ASSIGNINITIALPATH}()$ 
3   $\phi_f \leftarrow 0$ 
4   $P_{f,0}[\beta][v] \leftarrow P(v|y_\beta), \quad 0 \leq \beta < n, v \in \{0, 1\}$ 
5  PUSH( $0, f$ );  $\tau \leftarrow 1$ ;  $q \leftarrow (0, \dots, 0)$ 
6  while ( $\tau > 0$ )
7    ( $M, f$ )  $\leftarrow$  POPMAX();  $\tau \leftarrow \tau - 1$ 
8     $q_{\phi_f} \leftarrow q_{\phi_f} + 1$ 
9    if ( $\phi_f = n$ ) return ( $C_{f,0}[0, 0], \dots, C_{f,0}[n - 1, 0]$ )
10   RECURSIVELYCALCP( $f, m, \phi_f$ )
11   if ( $\phi_f \in \mathcal{F}$ )
12      $C_{f,m}[0, \phi_f \bmod 2] \leftarrow \text{FROZENVALUE}(f, \phi_f)$ 
13     PUSH( $P_{f,m}[0][0]\psi(\phi_f), f$ );  $\tau \leftarrow \tau + 1$ 
14     if ( $\phi_f$  odd) RECURSIVEUPDATEC( $f, m, \phi_f$ )
15      $\phi_f \leftarrow \phi_f + 1$ 
16   else
17     while ( $\tau \geq \Theta - 1$ )
18       ( $M, f$ )  $\leftarrow$  POPMIN()
19       KILLPATH( $f$ );  $\tau \leftarrow \tau - 1$ 
20        $C_{f,m}[0, \phi_f \bmod 2] \leftarrow 0$ 
21        $f' \leftarrow \text{CLONEPATH}(f)$ 
22        $C_{f',m}[0, \phi_f \bmod 2] \leftarrow 1$ 
23       PUSH( $P_{f,m}[0][0]\psi(\phi_f), f$ )
24       PUSH( $P_{f,m}[0][1]\psi(\phi_f), f$ );  $\tau \leftarrow \tau + 2$ 
25       if ( $\phi_f$  odd)
26         RECURSIVEUPDATEC( $f, m, \phi_f$ )
27         RECURSIVEUPDATEC( $f', m, \phi_{f'}$ )
28        $\phi_{f'} \leftarrow \phi_f \leftarrow \phi_f + 1$ 
29     if ( $q_{\phi_f} \geq L$ )
30       for каждого пути  $f'$  в стеке
31         if ( $\phi_{f'} < \phi_f$ )
32           ERASE( $f'$ ); KILLPATH( $f'$ );  $\tau \leftarrow \tau - 1$ 

```

Рис. 3.1. Декодирование методом направленного поиска

нее вероятные пути удаляются из очереди. В алгоритме используется процедура  $RecursiveUpdateC(f, \lambda, \phi)$ , совпадающая с  $RecursivelyUpdateC(\lambda, \phi)$ , приведенной на Рис. 1.8, за исключением того, что она выполняется только для одного пути с номером  $f$ . Приоритетная очередь, в которой поддерживаются требуемые операции, может быть реализована с использованием красно-черного дерева [37].

### 3.1.2. Численные результаты

Рис. 3.2 показывает среднее число итераций, выполняемых при декодировании. Представлены данные как для предлагаемого подхода, так и для стекового алгоритма последовательного исключения. Заметим, что использование метода направленного поиска позволяет значительно снизить число итераций.

На Рис. 3.3 представлены вычисленные для правильного пути  $u_0^{n-1}$  значения  $-\log P(u_0^{i-1}|y_0^{n-1}) - \log \phi(i)$  для нескольких запусков декодера, при этом значения  $\phi(i)$  получены с помощью Гауссовской аппроксимации (см. раздел 1.3.1.2). Если бы  $\phi(i)$  являлось точным значением вероятности еще непостроенной части правильного пути на фазе  $i$ , то значение  $P(u_0^{i-1}|y_0^{n-1})\phi(i)$  оставалось бы постоянным. При этом, результаты моделирования показывают, что корректирующая способность предлагаемого алгоритма декодирования оказывается такой же, как у списочного алгоритма последовательного исключения с аналогичным параметром  $L$ , в котором не используется эвристическая функция.

### 3.1.3. Выводы

В данном разделе представлен алгоритм направленного поиска для декодирования полярных кодов с ядром Арикана. Предлагаемый алгоритм построен на базе стекового алгоритма последовательного исключения, описанного в разделе 1.3.3, и обеспечивает существенное снижение сложности по сравнению со стековым алгоритмом, при той же корректирующей способности.

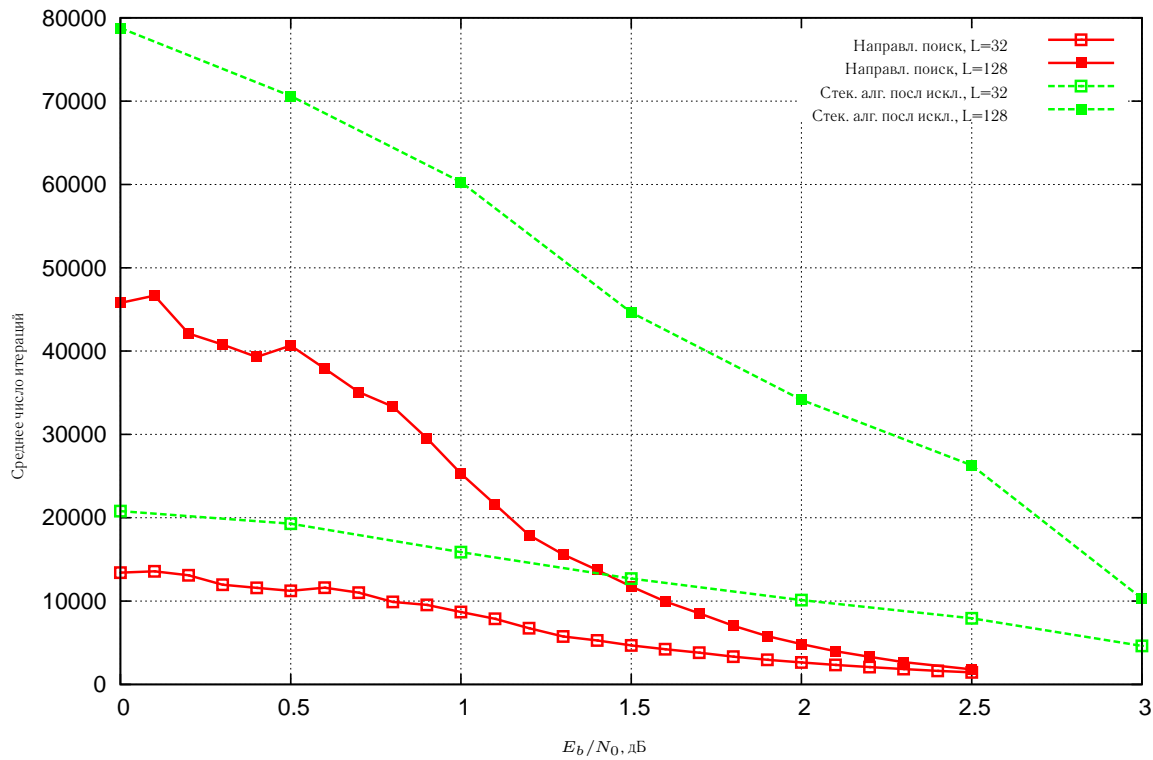


Рис. 3.2. Среднее число итераций

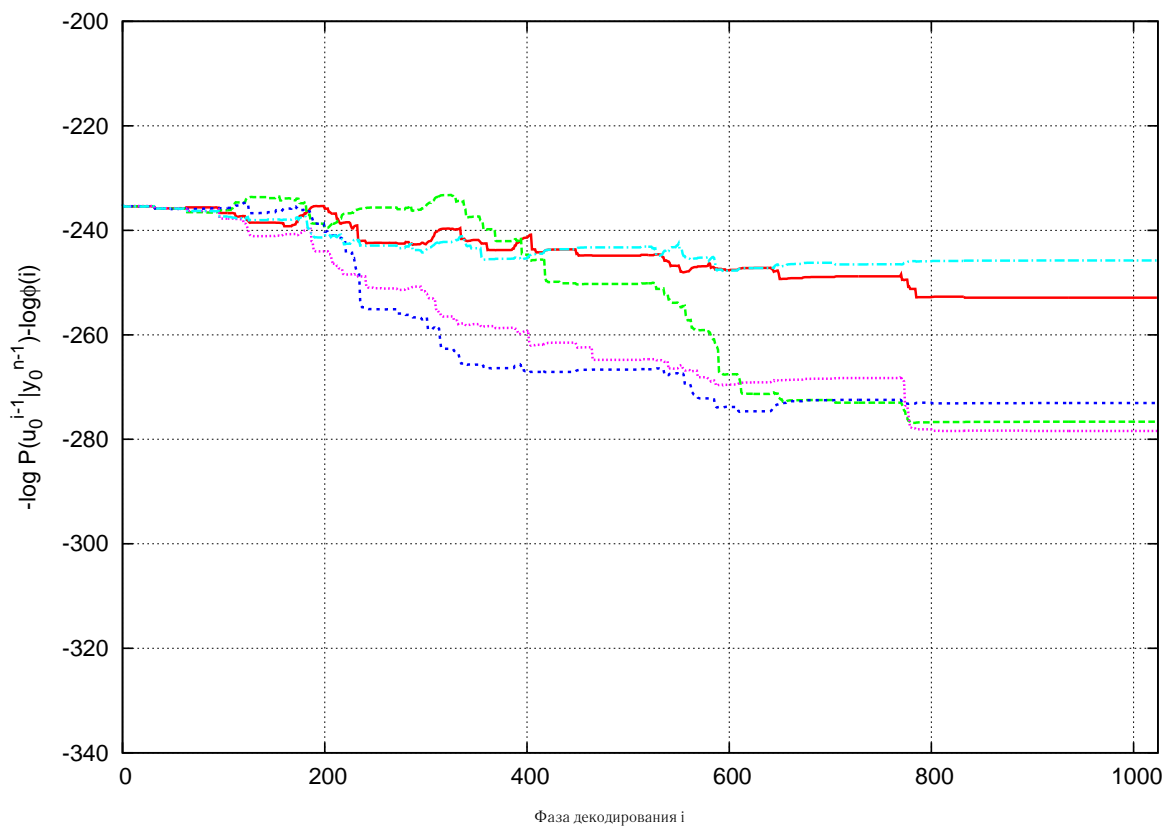


Рис. 3.3. Метрика пути

## 3.2. Последовательное декодирование полярных кодов с ядром Арикана

Цель декодера заключается в нахождении пути  $u_0^{n-1}, u_{0,\mathcal{F}}^{n-1} = 0$ , на котором достигается наибольшее значение вероятности  $P(u_0^{n-1}|y_0^{n-1})$ . Для построения эффективного декодера на базе стекового алгоритма последовательного исключения, выполняющего поэлементное построение путей, необходимо разработать надежный метод предсказания того, какое начало пути  $u_0^i$  может соответствовать решению задачи декодирования.

### 3.2.1. Метрика пути

Оценим вероятность  $T(u_0^i, y_0^{n-1}) = \max_{u_{i+1}^{n-1}: u_{i+1,\mathcal{F}}^{n-1} = 0} P(u_0^{n-1}|y_0^{n-1})$  передачи наиболее вероятного кодового слова полярного кода, соответствующего входной последовательности с началом  $u_0^i$ . Эта оценка может быть использована для того, чтобы предсказать, какой путь  $u_0^i$  может соответствовать решению задачи декодирования.

Вычисление вероятности  $T(u_0^i, y_0^{n-1})$  на  $i$ -ой фазе алгоритма последовательного исключения является крайне трудоемкой задачей. Рассмотрим  $2^{n-i-1}$  различных путей  $v[j]_0^{n-1}$  таких, что  $v[j]_0^i = u_0^i$  и  $v[j]_{i+1}^{n-1} \in \{0, 1\}^{n-i-1}$ ,  $0 \leq j < 2^{n-i-1}$ . Предположим, что  $u_0^i$  соответствует наиболее вероятному кодовому слову. Пусть  $J$  – случайная величина, равная  $j$ , если наиболее вероятное кодовое слово полярного кода соответствует пути  $v[j]_0^{n-1}$ . Заметим, что  $J = j$  предполагает  $v[j]_h = 0$ ,  $h \in \mathcal{F}$ . Мы предлагаем оценить  $T(u_0^i, y_0^{n-1})$  как

$$\begin{aligned} T(u_0^i, y_0^{n-1}) &\approx E_J[P(v[J]_0^{n-1}|y_0^{n-1})] = \\ &= \sum_{j=0}^{2^{n-i-1}-1} P(v[j]_0^{n-1}|y_0^{n-1})P\{J = j\} \geq \underbrace{P(v[\alpha]_0^{n-1}|y_0^{n-1})}_{R(u_0^i, y_0^{n-1})} P\{J = \alpha\}, \end{aligned} \quad (3.1)$$

где  $\alpha = \arg \max_{0 \leq j < 2^{n-i-1}} P(v[j]_0^{n-1}|y_0^{n-1})$ .

При возрастании  $i$  происходит уменьшение числа возможных значений  $J$ , что приводит к уточнению вычисляемой оценки. При усреднении  $P \{J = \alpha\}$  по всем принятым последовательностям получаем вероятность того, что наиболее вероятный путь  $u_0^{n-1}$  с началом  $u_0^i$  имеет нули на позициях  $j \in \mathcal{F}$ . Эта вероятность ограничена снизу вероятностью  $\hat{\Omega}(i)$  того, что декодер последовательного исключения, начинающий работу с  $u_0^i$  и не принимающий во внимание условия заморозки, примет решения  $u_j = 0, j > i, j \in \mathcal{F}$ , то есть не сделает ошибок при принятии решений относительно значений замороженных символов.

Таким образом, получаем следующую оценку для  $T(u_0^i, y_0^{n-1})$ :

$$\hat{T}(u_0^i, y_0^{n-1}) = R(u_0^i, y_0^{n-1})\hat{\Omega}(i) \quad (3.2)$$

где  $\hat{\Omega}(i) = \prod_{j \in \mathcal{F}, j > i} (1 - P_j)$ . Задача вычисления вероятностей  $P_j$  рассматривается в разделе 1.3.1.2. Логарифм величины  $\hat{T}(u_0^i, y_0^{n-1})$  предлагается использовать в качестве метрики пути. Значение  $\hat{\Omega}(i)$  зависит только от длины кода  $n$ , множества замороженных символов  $\mathcal{F}$  (то есть от характеристик полярного кода), свойств канала и фазы  $i$ .

На каждой итерации стекового алгоритма декодирования увеличивается длина пути с наибольшим значением  $\hat{T}(u_0^i, y_0^{n-1})$ . Заметим, что для любого пути  $u_0^i$  значение  $\hat{\Omega}(i)$  возрастает с  $i$ , в то время как  $R(u_0^i, y_0^{n-1})$  убывает с  $i$ . В том случае, если два пути имеют одинаковые значения вероятностей  $R(u_0^i, y_0^{n-1})$ , декодер выберет путь большей длины. Данный подход позволяет сравнивать пути  $u_0^i$  различных длин, предотвращая при этом частое переключение между различными путями.

Значения вероятностей  $R(u_0^i, y_0^{n-1})$  могут быть вычислены с использованием свойств кодера полярных кодов. Операция кодирования  $c_0^{n-1} = u_0^{n-1}G_n$  может быть представлена как рекурсивное применение выражений  $c_0^{n/2-1} = (u_{0,E}^{n-1} \oplus u_{0,O}^{n-1})G_{n/2}$  и  $c_{n/2}^{n-1} = u_{0,O}^{n-1}G_{n/2}$ . Напомним, что  $E$  и  $O$  являются множествами всех четных и нечетных чисел, соответственно. То есть, кодирование входной последовательности длины  $n$  сводится к кодированию последовательностей

$(u_{0,E}^{n-1} \oplus u_{0,O}^{n-1})$  и  $u_{0,O}^{n-1}$  длины  $n/2$ , что предполагает

$$R(u_0^{2i}, y_0^{n-1}) = \max_{u_{2i+1} \in \{0,1\}} R(u_{0,E}^{2i+1} \oplus u_{0,O}^{2i+1}, y_0^{n/2-1}) R(u_{0,O}^{2i+1}, y_{n/2}^{n-1}), \quad (3.3)$$

$$R(u_0^{2i+1}, y_0^{n-1}) = R(u_{0,E}^{2i+1} \oplus u_{0,O}^{2i+1}, y_0^{n/2-1}) R(u_{0,O}^{2i+1}, y_{n/2}^{n-1}). \quad (3.4)$$

Начальные значения для данных рекурсивных выражений задаются  $R(b, y_j) = P(b|y_j)$ ,  $b \in \{0, 1\}$ .

### 3.2.2. Эффективная реализация

Эффективная реализация предлагаемого подхода может быть построена на основе реализации списочного алгоритма последовательного исключения, описанной в разделе 1.3.4. Пусть  $\lambda$ ,  $\phi$  и  $\beta$  означают слой, фазу и номер ветви, соответственно, где  $0 \leq \beta < 2^{m-\lambda}$ . На каждой итерации декодирования необходимо вычислить только  $R((u_0^i, b), y_0^{n-1})$ ,  $b \in \{0, 1\}$ , исходя из чего,  $R((u_0^i, b), y_0^{n-1})$  можно сопоставить номер ветви  $\beta = 0$ . Само значение  $R((u_0^i, b), y_0^{n-1})$  может быть рекурсивно вычислено следующим образом. Для  $\lambda > 0$  следует вычислить величину  $R(\hat{u}_0^{\phi-1}, y_0^{\Lambda-1})$ , сопоставим ей номер ветви  $\beta$ , где  $\Lambda = 2^\lambda$ . Обозначим  $\psi = \lfloor \phi/2 \rfloor$ . Тогда  $R(\hat{u}_{0,E}^{2\psi-1} \oplus \hat{u}_{0,O}^{2\psi-1}, y_0^{\Lambda/2-1})$  и  $R(\hat{u}_{0,O}^{2\psi-1}, y_{\Lambda/2}^{\Lambda-1})$  связаны с номерами ветвей  $2\beta$  и  $2\beta + 1$ , соответственно.

Пусть  $\langle \phi, \beta \rangle_\lambda = \phi + 2^\lambda \beta$  и

$$R_\lambda[\langle \phi, \beta \rangle_\lambda][b] = R((\hat{u}_0^{\phi-1}, b), y_0^{\Lambda-1}).$$

Аналогично 1.3.4, мы может отбросить индекс  $\phi$ . Для краткости эту величину мы будем обозначать  $R_\lambda[\beta][b]$ .

При использовании представления  $R_\lambda$  в виде массива, выражения (3.3) и (3.4) преобразуются в

$$R_\lambda[\beta][b] = \begin{cases} \max_{d \in \{0,1\}} R_{\lambda-1}[2\beta][b \oplus d] R_{\lambda-1}[2\beta + 1][d], & \phi \in E, \\ R_{\lambda-1}[2\beta][C_\lambda[\beta][0] \oplus b] R_{\lambda-1}[2\beta + 1][b], & \phi \in O, \end{cases}$$



где массив  $C_\lambda[\beta][b]$  определен как в разделе 1.3.4.

После выполнения замены переменных  $S_\lambda[\beta] = \ln \frac{R_\lambda[\beta][0]}{R_\lambda[\beta][1]}$  получаем, что

$$S_\lambda[\beta] = \begin{cases} \max(S_{\lambda-1}[2\beta] + S_{\lambda-1}[2\beta + 1], 0) - \\ \quad \max(S_{\lambda-1}[2\beta], S_{\lambda-1}[2\beta + 1]), & \phi \in E, \\ (-1)^{C_\lambda[\beta][0]} S_{\lambda-1}[2\beta] + S_{\lambda-1}[2\beta + 1], & \phi \in O. \end{cases} \quad (3.5)$$

$$R_\lambda[\beta][1] = R_{\lambda-1}[2\beta][1] R_{\lambda-1}[2\beta + 1][1] \exp(Z_\lambda[\beta]),$$

где

$$Z_\lambda[\beta] = \begin{cases} \max(S_{\lambda-1}[2\beta], S_{\lambda-1}[2\beta + 1]), & \phi \in E, \\ C_\lambda[\beta][0] S_{\lambda-1}[2\beta], & \phi \in O. \end{cases} \quad (3.6)$$

Заметим, что

$$\ln R_m[0][1] = \left( \sum_{j=0}^{2^m-1} \ln P(1|y_j) \right) + \left( \sum_{\lambda=1}^m \sum_{\beta=0}^{2^{m-\lambda}-1} Z_\lambda[\beta] \right),$$

следовательно, значения  $D_m = \ln R_m[0][1]$  могут быть вычислены следующим образом. Пусть  $D_0 = \sum_{j=0}^{2^m-1} \ln P(1|y_j)$ . Для  $1 \leq \lambda \leq m$  получаем

$$D_\lambda = D_{\lambda-1} + \sum_{j=0}^{2^{m-\lambda}-1} Z_\lambda[j]. \quad (3.7)$$

В силу того, что  $\ln R_m[0][1] = D_m$ ,  $\ln R_m[0][0] = D_m + S_m[0]$  и значение  $D_0$  не зависит от  $u_0^i$ , мы можем положить  $D_0 = 0$  в данной рекурсии, при этом значения  $\ln R_m[0][b]$ ,  $b \in \{0, 1\}$ , изменятся на одну и ту же величину для всех путей  $u_0^i$ , что не повлияет на порядок путей в приоритетной очереди.

Описанные выше вычисления могут быть реализованы следующим образом. Пусть  $f$  — индекс пути  $u_0^{\phi_f-1}$ . Для каждого  $f$  необходимо хранить значения  $S_{f,\lambda}[\beta]$ ,  $D_{f,\lambda}$ , а также  $C_{f,\lambda}[\beta][b]$ ,  $0 \leq \lambda \leq m$ ,  $0 \leq \beta < 2^{m-\lambda}$ ,  $b \in \{0, 1\}$ . Алгоритм *RecursivelyCalcS*, представленный на Рис. 3.4, реализует вычисления, задаваемые выражениями (3.5)–(3.7). При вызове *RecursivelyCalcS*( $f, m, \phi_f$ ) переиспользуются значения, полученные при предыдущих вызовах *RecursivelyCalcS* для того же значения  $f$ .

RECURSIVELYCALCS( $f, \lambda, \phi$ )

```
1  if ( $\lambda = 0$ ) return
2  if ( $\phi$  even) RECURSIVELYCALCS( $f, \lambda - 1, \lfloor \phi/2 \rfloor$ )
3   $D_{f,\lambda} \leftarrow D_{f,\lambda-1}$ 
4  for ( $\beta = 0, \dots, 2^{m-\lambda} - 1$ )
5     $t \leftarrow S_{f,\lambda-1}[2\beta]$ 
6     $t' \leftarrow S_{f,\lambda-1}[2\beta + 1]$ 
7    if ( $\phi$  even)
8      if ( $t < t'$ )
9        if ( $t > -t'$ )  $S_{f,\lambda}[\beta] \leftarrow t$  else  $S_{f,\lambda}[\beta] \leftarrow -t'$ 
10        $D_{f,\lambda} \leftarrow D_{f,\lambda} + t'$ 
11      else
12        if ( $t > -t'$ )  $S_{f,\lambda}[\beta] \leftarrow t'$  else  $S_{f,\lambda}[\beta] \leftarrow -t$ 
13        $D_{f,\lambda} \leftarrow D_{f,\lambda} + t$ 
14      else
15        if ( $C_{f,\lambda}[\beta][0] = 0$ )
16          $S_{f,\lambda}[\beta] \leftarrow t + t'$ 
17        else
18          $S_{f,\lambda}[\beta] \leftarrow t' - t$ 
19          $D_{f,\lambda} \leftarrow D_{f,\lambda} + t$ 
```

Рис. 3.4. Вычисление метрики пути

На Рис. 3.5 представлен предлагаемый алгоритм последовательного декодирования. Инициализация отличается от описанной в разделе 1.3.4 только тем, что массив вероятностей  $P$  заменяется массивом  $S$  и создается дополнительный массив  $D$ . Для этих массивов используется тот же метод ленивого копирования структур данных, что и в списочном алгоритме последовательного исключения Тала-Варди. Элементы  $S_{f,0}[j]$ ,  $0 \leq j < n$ , инициализируются логарифмическими отношениями правдоподобия  $\ln(P(0|y_j))/(P(1|y_j))$ . Пути в стеке (приоритетной очереди) упорядочены согласно своим метрикам. На каждой итерации путь с наибольшей метрикой  $M$  извлекается из стека. Для этого пути вычисляется логарифмическое отношение правдоподобия  $S_{f,m}[0]$  и логарифм вероятности  $D_{f,m}$ . Если фаза  $\phi_f$  соответствует замороженному символу, то для расширения пути используется только значение  $FrozenValue(f, \phi_f)$ , задаваемое ограничением статической/динамической заморозки (2.1). В противном случае, путь копируется и рассматриваются оба варианта значения  $u_i \in \{0, 1\}$ . Метрики расширенных путей включают слагаемое  $\hat{\Omega}_{ln}(\phi) = \ln \hat{\Omega}(\phi)$ . В алгоритме используется процедура  $RecursiveUpdateC(f, \lambda, \phi)$ , совпадающая с  $RecursiveUpdateC(\lambda, \phi)$ , приведенной на Рис. 1.8, за исключением того, что она выполняется для заданного  $f$ .

Чтобы предотвратить переполнение приоритетной очереди  $\Theta$ , пути с наименьшими метриками исключаются из очереди на строке 19. Если число различных путей, построенных до фазы  $q_{\phi_f}$ , превышает  $L$ , то короткие пути исключаются из очереди на строке 32.

Заметим, что предлагаемый алгоритм использует только операции сложения, вычитания и сравнения. Кроме того, массив  $P_{f,\lambda}[\beta][b]$ , используемый в алгоритме Тала-Варди, заменяется на массив в два раза меньшего размера  $S_{f,\lambda}[\beta]$ .

Сложность декодирования может быть дополнительно снижена за счет использования следующих приемов:

- Пусть на фазе  $(\phi_f - 1)$  были вычислены значения  $a_0 = S_{f,m}[0] + D_{f,m}$  и

```

DECODE( $y_0^{n-1}, L, \Theta$ )
1  INITIALIZEDATASTRUCTURES()
2   $f \leftarrow \text{ASSIGNINITIALPATH}(); \phi_f \leftarrow 0$ 
3   $S_{f,0}[\beta] \leftarrow \ln \frac{P(0|y_\beta)}{P(1|y_\beta)}, \beta = 0, \dots, n-1$ 
4   $D_{f,0} \leftarrow 0$ 
5  PUSH( $0, f$ );  $P \leftarrow 1; q \leftarrow (0, \dots, 0)$ 
6  while ( $P > 0$ )
7    ( $M, f$ )  $\leftarrow$  POPMAX();  $P \leftarrow P - 1$ 
8     $q_{\phi_f} \leftarrow q_{\phi_f} + 1$ 
9    if ( $\phi_f = n$ ) return ( $C_{f,0}[0, 0], \dots, C_{f,0}[n-1, 0]$ )
10   RECURSIVELYCALCS( $f, m, \phi_f$ )
11   if ( $\phi_f \in \mathcal{F}$ )
12      $C_{f,m}[0, \phi_f \bmod 2] \leftarrow \text{FROZENVALUE}(f, \phi_f)$ 
13     PUSH( $S_{f,m}[0] + D_{f,m} + \hat{\Omega}_{ln}(\phi_f), f$ );  $P \leftarrow P + 1$ 
14     if ( $\phi_f$  odd) RECURSIVEUPDATEC( $f, m, \phi_f$ )
15      $\phi_f \leftarrow \phi_f + 1$ 
16   else
17     while ( $P \geq \Theta - 1$ )
18       ( $M, f$ )  $\leftarrow$  POPMIN()
19       KILLPATH( $f$ );  $P \leftarrow P - 1$ 
20        $C_{f,m}[0, \phi_f \bmod 2] \leftarrow 0$ 
21        $f' \leftarrow \text{CLONEPATH}(f)$ 
22        $C_{f',m}[0, \phi_f \bmod 2] \leftarrow 1$ 
23       PUSH( $S_{f,m}[0] + D_{f,m} + \hat{\Omega}_{ln}(\phi_f), f$ )
24       PUSH( $D_{f,m} + \hat{\Omega}_{ln}(\phi_f), f'$ );  $P \leftarrow P + 2$ 
25       if ( $\phi_f$  odd)
26         RECURSIVEUPDATEC( $f, m, \phi_f$ )
27         RECURSIVEUPDATEC( $f', m, \phi_f$ )
28        $\phi_{f'} \leftarrow \phi_f \leftarrow \phi_f + 1$ 
29   if ( $q_{\phi_f} \geq L$ )
30     for каждого пути  $f'$  в стеке
31       if ( $\phi_{f'} < \phi_f$ )
32         ERASE( $f'$ ); KILLPATH( $f'$ );  $P \leftarrow P - 1$ 

```

Рис. 3.5. Последовательное декодирование полярных кодов

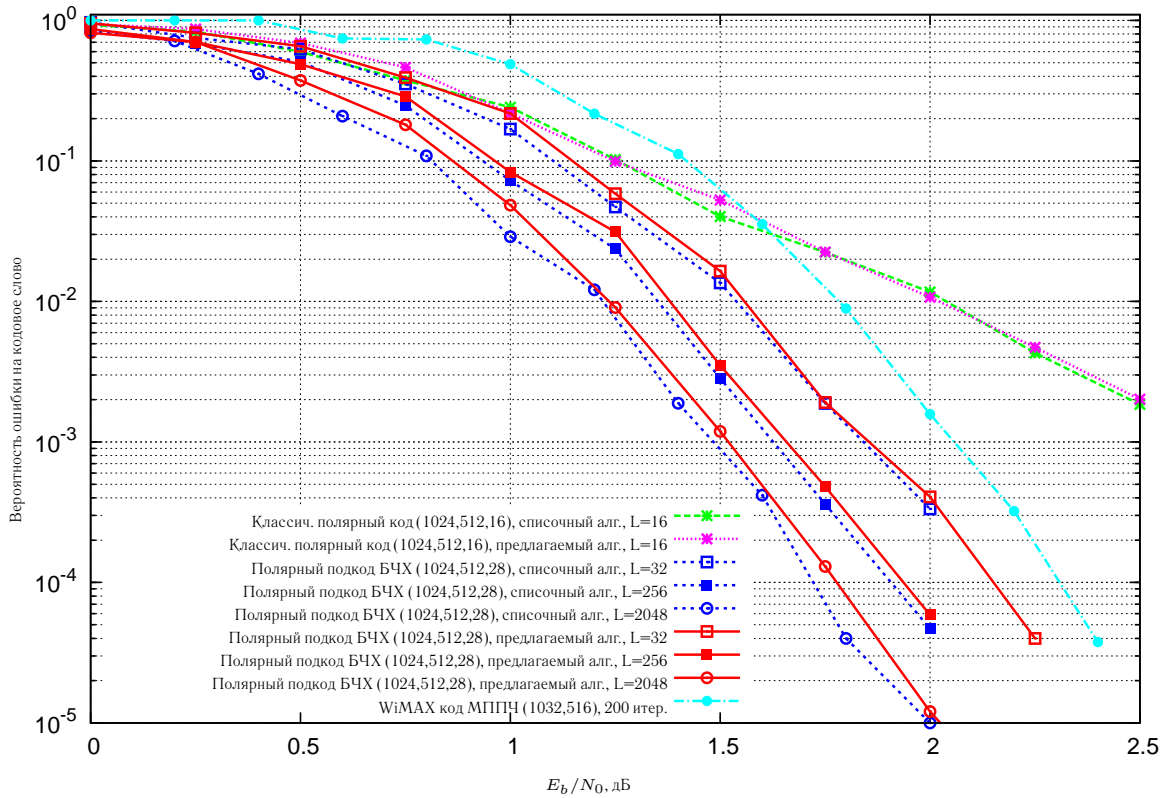


Рис. 3.6. Вероятность ошибки декодирования (1024, 512) полярных кодов

$a_1 = D_{f,m}$ . Тогда на фазе  $\phi_f$  достаточно вычислить только  $a'_0 - a'_1 = S'_{f,m}[0]$ , поскольку всегда существуют такие  $b, t \in \{0, 1\}$ , что выполняются равенства  $a'_{1-t} = a_b$  и  $a'_t = a_b + (-1)^t S'_{f,m}[0]$ .

- Функцию *RecursivelyCalcS* можно вызывать только для  $\phi_f \notin \mathcal{F}$ . При удлинении пути на замороженный символ сохраняется прежнее значение метрики. Заметим, что это требует некоторых изменений в *RecursivelyCalcS*, поскольку необходимо обеспечить доступность данных для вычислений.
- Если все замороженные символы уже обработаны, то декодер может перейти к жесткому декодированию.

### 3.2.3. Численные результаты

На Рис. 3.6 представлен график зависимости вероятности ошибки декодирования от отношения сигнал/шум для предлагаемого алгоритма, списочного алгоритма Тала-Варди (см. раздел 1.3.2) и алгоритма декодирования с направленным поиском (см. раздел 3.1). Результаты приведены как для (1024, 512, 16) классического полярного кода и (1024, 512, 28) полярного подкода, так и для (1032, 516) кода МППЧ. Из графика видно, что корректирующая способность предлагаемого алгоритма декодирования в случае классического полярного кода совпадает с таковой списочного алгоритма Тала-Варди. Необходимо заметить, что корректирующая способность этого кода ограничена малым минимальным расстоянием, и увеличение размера списка  $L$  не приводит к ее улучшению. Код (1024, 512, 28) обладает существенно большей корректирующей способностью. Видно, что вероятность ошибки декодирования с помощью предлагаемого алгоритма незначительно выше вероятности ошибки декодирования при использовании метода направленного поиска с аналогичным параметром  $L$ .

На Рис. 3.7 приведены значения метрики правильного пути  $\ln \hat{T}(u_0^i, y_0^{n-1})$  (см. выражение (3.2)) для нескольких запусков декодера. Видно, что получаемые значения метрики близки к итоговому значению  $\ln P(u_0^{n-1} | y_0^{n-1})$  уже после того, как декодер обработает первый блок замороженных символов (фазы, соответствующие замороженным символам помечены +). Это препятствует переключению декодера на неправильный путь.

Среднее число итераций цикла *WHILE*, выполняемое предлагаемым алгоритмом (см. Рис. 3.5) и методом направленного поиска, представлено на Рис. 3.8. Видно, что предлагаемый алгоритм обладает существенно меньшей сложностью. Это позволяет выбрать больший размер списка  $L$ , обеспечивающий меньшую вероятность ошибки декодирования, при сохранении вычислительной сложности на уровне метода направленного поиска. К примеру, для отношения сигнал/шум 1.75 дБ среднее число итераций, выполняемых предлагаемым алго-

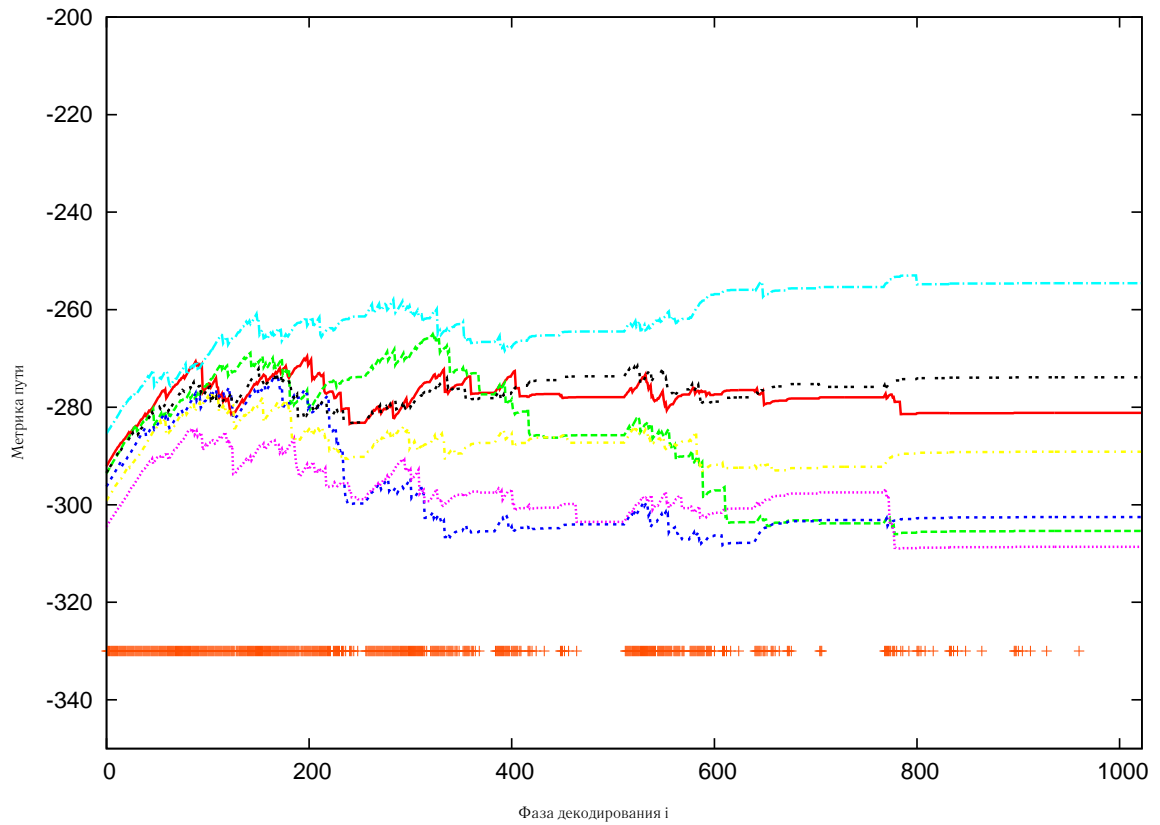


Рис. 3.7. Метрика правильного пути для (1024, 512, 28) полярного подкода

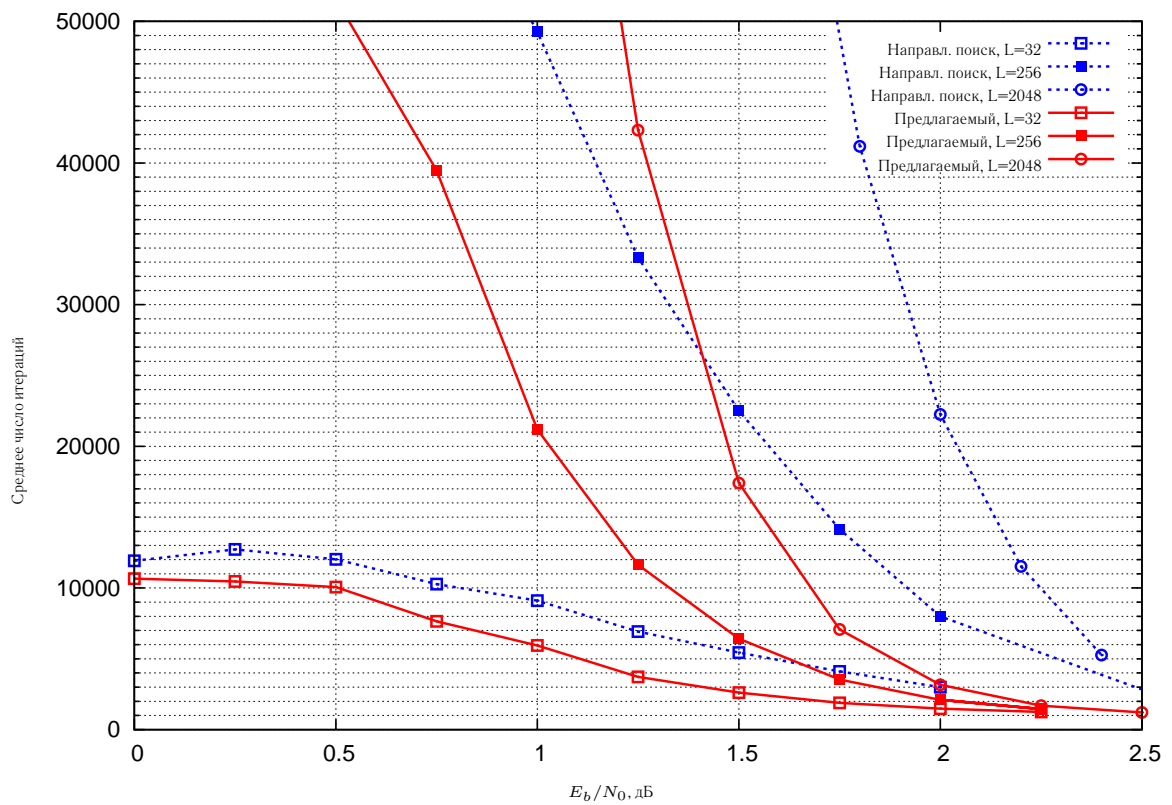


Рис. 3.8. Среднее число итераций при декодировании (1024, 512, 28) полярного подкода

Таблица 3.1. Средняя сложность декодирования,  $\times 10^3$  вещественных операций

$E_b/N_0$ , дБ	Предлагаемый подход						Распр. дов., МППЧ	
	Сложения			Сравнения			Сложения	$\ln \tanh(\frac{x}{2})$
	$L$			$L$			итер.	итер.
	32	256	2048	32	256	2048	$\leq 200$	$\leq 200$
0	141	833	5231	227	1332	8374	2617	1307
0.25	145	866	5321	236	1398	8601	2617	1307
0.5	133	752	4265	218	1224	6968	2333	1112
0.75	105	524	2584	173	862	4271	2041	1003
1	73	286	1232	122	477	2065	1469	722
1.25	47	158	601	79	268	1022	711	345
1.5	32	88	267	54	151	461	394	185
1.75	23	49	107	40	86	186	205	94
2	18	27	42	31	48	74	140	62
2.25	14	17	18	25	31	32	115	50

ритмом при  $L = 256$ , совпадает с таковым, выполняемым методом направленного поиска при  $L = 32$ . Заметим, что в наихудшем случае сложность предлагаемого алгоритма равна  $O(Ln \log(n))$  операций над вещественными числами и совпадает со сложностью списочного алгоритма последовательного исключения Тала-Варди с аналогичным размером списка  $L$ . При этом средняя сложность предлагаемого алгоритма оказывается существенно ниже сложности списочного алгоритма Тала-Варди. Кроме того, каждая итерация предлагаемого алгоритма содержит только операции сложения и сравнения, таким образом, его сложность реализации существенно ниже сложности реализации списочного/стекового алгоритма последовательного исключения, которые требуют выполнения умножений или вычисления логарифмов Якоби (в том случае, если алгоритм оперирует логарифмами вероятностей).

Таблица 3.1 иллюстрирует число операций над вещественными числами, выполняемых при декодировании полярного подкода с использованием предлагаемого подхода, а также кода МППЧ при декодировании с помощью алгоритма



распространения доверия с не более чем 200 итерациями. Видно, что предлагаемый подход требует выполнения меньшего числа операций сложения.

### 3.2.4. Выводы

В данном разделе был представлен алгоритм последовательного декодирования полярных кодов, основанный на стековом алгоритме последовательного исключения. Предлагаемый алгоритм осуществляет обход кодового дерева в зависимости от вычисленных оценок апостериорных вероятностей кодовых слов. Результаты моделирования показывают, что по сравнению со стековым алгоритмом последовательного исключения и методом направленного поиска (см. раздел 3.1), предлагаемый подход обеспечивает существенное снижение сложности при незначительном ухудшении корректирующей способности. При этом вычислительная сложность при декодировании полярных кодов с помощью предлагаемого алгоритма меньше, чем при декодировании МППЧ кодов с помощью алгоритма распространения доверия со схожими параметрами.

## 3.3. Алгоритм последовательного исключения для полярных кодов с произвольным двоичным ядром

Декодирование полярного кода с  $l \times l$  ядром  $M$  с помощью алгоритма последовательного исключения, изложенного в разделе 1.3.1.1, сводится к рекурсивному вычислению ЛОПП  $L_{u_i} = \log \frac{P(\hat{u}_0^{i-1}, u_i=0|y_0^{l-1})}{P(\hat{u}_0^{i-1}, u_i=1|y_0^{l-1})}$  для заданных значений символов  $\hat{u}_0^{i-1}$ ,  $0 \leq i < l$ . Исходными данными для вычисления  $L_{u_i}$  являются ЛОПП принятых символов  $L_i = \ln \frac{P(0|y_i)}{P(1|y_i)}$ ,  $0 \leq i < l$ , или ЛОПП, вычисленные при декодировании на предыдущем слое поляризующего преобразования.

Воспользуемся следующим представлением ЛОПП

$$\begin{aligned}
L_{u_i} &= \ln \frac{\sum_{u_{i+1}^{l-1}} P(\hat{u}_0^{i-1}, u_i = 0, u_{i+1}^{l-1} | y_0^{l-1})}{\sum_{u_{i+1}^{l-1}} P(\hat{u}_0^{i-1}, u_i = 1, u_{i+1}^{l-1} | y_0^{l-1})} = \\
&= \ln \frac{\sum_{c_0^{l-1} \in C[0]} \prod_{j=0}^{l-1} P(c_j | y_j)}{\sum_{c_0^{l-1} \in C[1]} \prod_{j=0}^{l-1} P(c_j | y_j)} = \\
&= \ln \frac{\sum_{c_0^{l-1} \in C[0]} \exp(\text{corr}(c_0^{l-1}, L_0^{l-1}) / 2)}{\sum_{c_0^{l-1} \in C[1]} \exp(\text{corr}(c_0^{l-1}, L_0^{l-1}) / 2)}, \tag{3.8}
\end{aligned}$$

где множество  $C[b]$  состоит из кодовых слов  $c_0^{l-1} = (\hat{u}_0^{i-1}, b, u_{i+1}^{l-1})M$ ,  $u_{i+1}^{l-1} \in \{0, 1\}^{l-i-1}$ , корреляция  $\text{corr}(c_0^{l-1}, L_0^{l-1}) = \sum_{i=0}^{l-1} (-1)^{c_i} L_i$ . Вычислительная сложность полного перебора всех кодовых слов  $c_0^{l-1}$  для числителя и знаменателя выражения (3.8) оказывается чрезвычайно высокой. Далее рассматривается два метода вычисления  $L_{u_i}$ .

### 3.3.1. Точный метод

Как числитель, так и знаменатель выражения (3.8) равны сумме вероятностей передачи всех кодовых слов некоторого смежного класса кода  $C_i$ , порожденного последними  $(l - i - 1)$  строками ядра  $M$ . Такая сумма вероятностей может быть найдена путем обхода решетки кода  $C_i$ . Каждый узел этой решетки характеризуется парой чисел  $(x, v)$ , где  $x$  равно количеству обработанных столбцов проверочной матрицы  $H$  кода  $C_i$ ,  $0 \leq x \leq l$ , число  $v$  в двоичном представлении равно соответствующей линейной комбинации первых  $x$  столбцов матрицы  $H$ . Дуга, соединяющая два соседних узла  $(x - 1, v)$  и  $(x, v')$ , при  $v = v'$  соответствует значению  $\omega(x, v, v') = 0$ , иначе  $\omega(x, v, v') = 1$ . Каждый путь из узла  $(0, 0)$  в  $(l, 0)$  задает кодовое слово,  $x$ -ый бит которого равен  $\omega(x, v, v')$ . На Рис. 3.9 приведен пример такой решетки для кода БЧХ (7, 4).

Сумма в числителе или знаменателе выражения (3.8) может быть вычислена следующим образом. Пусть ЛОПП  $L_x^{(b)} = (-1)^{(u_0^{i-1}, b)M_{0..i, x}} L_x$ , где  $M_{0..i, x}$  — вектор, состоящий из  $0, \dots, i$  элементов  $x$ -го столбца матрицы  $M$ ,  $b \in \{0, 1\}$ . В

$$H = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

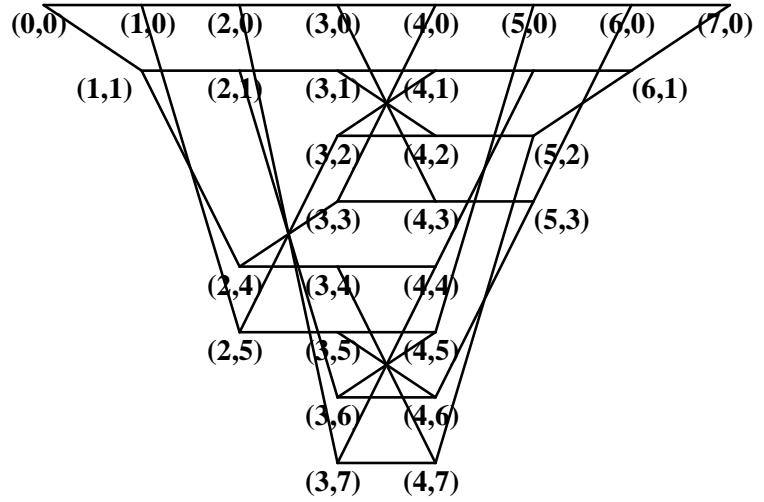


Рис. 3.9. Решетка для (7, 4) кода БЧХ

решетке для каждой дуги  $(x, v, v')$  вычислим  $\gamma^{(b)}(x, v, v') = \exp((-1)^{\omega(x,v,v')} L_x^{(b)})$ . Затем, для каждого узла на уровне  $1 \leq x \leq l$  найдем значение  $\sigma^{(b)}(x, v) = \sum_{v'} \sigma^{(b)}(x-1, v') \gamma^{(b)}(x, v', v)$ , где  $\sigma^{(b)}(0, 0) = 1$ . Значения  $\sigma^{(0)}(l, 0)$  и  $\sigma^{(1)}(l, 0)$  равны числителю и знаменателю выражения (3.8). Таким образом, ЛОПП  $L_{u_i}$  могут быть вычислены как  $L_{u_i} = \ln \frac{\sigma^{(0)}(l, 0)}{\sigma^{(1)}(l, 0)}$ . Заметим, что решетка кода служит для выявления общих сомножителей в слагаемых, участвующих в данных суммах. Сложность предлагаемого подхода зависит от профиля сложности решетки кода, она оказывается чрезвычайно высокой для больших  $l$  и  $i \approx l/2$ . Заметим также, что для  $i > 0$  достаточно вычислять только числитель или знаменатель дроби выражения (3.8), поскольку сумма числителя и знаменателя дроби для  $u_i$  равна числителю или знаменателю дроби для  $u_{i-1}$  (в зависимости от решения, принятого относительно значения  $u_{i-1}$ ). Таким образом, для всех  $i > 0$  для вычисления  $L_{u_i}$  достаточно одного прохода по решетке.

Существует множество алгоритмов декодирования, использующих представление кода в виде решетки, одним из них является алгоритм Бала-Кокке-Елинека-Равива, предложенный в [64]. Предлагаемый метод вычисления сумм числителя и знаменателя (3.8) можно рассматривать как применение прямого

хода алгоритма Бала-Коке-Елинека-Равива. ЛОПП  $L_{u_i}$  могут быть вычислены с помощью алгоритма, предложенного в [3], в основе которого также лежит алгоритм Бала-Коке-Елинека-Равива. Поскольку алгоритм Бала-Коке-Елинека-Равива позволяет вычислить только ЛОПП для символов кодового слова, то для нахождения ЛОПП  $L_{u_i}$  в [3] предлагается дополнить кодовые слова кода информационными символами  $u_i$ , то есть дополнить порождающую матрицу кода столбцами, состоящими из нулей и содержащими единицу лишь на  $i$ -ой позиции. Заметим, что при добавлении дополнительных столбцов к порождающей матрице профиль сложность соответствующей решетки увеличивается, в результате чего, возрастает сложность обхода этой решетки.

### 3.3.2. Декодирование с помощью порядковых статистик

Метод декодирования, основанный на порядковых статистиках, был предложен для случая двоичных линейных кодов в [32]. Данный метод позволяет исправить  $t$  ошибок на наиболее вероятной информационной совокупности, где порядок предобработки  $t$  является параметром алгоритма.

При вычислении сумм в числителе и знаменателе выражения (3.8) можно исключить малозначимые слагаемые, которые почти не оказывают влияния на значение  $L_{u_i}$ . Метод декодирования, основанный на порядковых статистиках позволяет перебрать  $\binom{l-i}{t}$  последовательностей  $(\hat{u}_0^{i-1}, u_i^{l-1})M$ , являющихся кодовыми словами смежного класса кода с порождающей матрицей  $M^{(i)}$ , состоящей из последних  $(l-i)$  строк матрицы  $M$ . Алгоритм осуществляет перебор всех кодовых слов, отличающихся не более чем на  $t$  позициях с наиболее вероятной информационной совокупностью, состоящей из  $(l-i)$  позиций наибольшим по модулю ЛОПП  $L_j$  при условии линейной независимости соответствующих столбцов матрицы  $M^{(i)}$ . Заметим, что для каждого найденного кодового слова  $c_0^{l-1}$  есть не менее  $(l-i-t)$  положительных  $(-1)^{c_j} L_j$ , соответствующих наиболее вероятной информационной совокупности, вследствие чего, кодовые слова

Таблица 3.2. Время декодирования, с

	По определению	ПС	ПС и ТМ
$l = 15, M^{\otimes 2}$	0.78	0.004	0.003
$l = 31, M^{\otimes 2}$	—	0.108	0.085

$c_0^{l-1}$  обладают большими корреляциями. Каждое кодовое слово участвует в сумме числителя или знаменателя, в зависимости от значения  $u_i$ .

### 3.3.3. Численные результаты

В результате статистического моделирования были получены оценки времени декодирования полярных кодов с ядром БЧХ (см. раздел 1.1.2), представленные в Таблице 3.2. Непосредственная реализация побитового декодирования согласно формуле (3.8), значительно проигрывает по сложности методу, основанному на порядковых статистиках “ПС”, и ему же в сочетании точным методом вычисления ЛОПП “ТМ”.

### 3.3.4. Выводы

В случае полярных кодов с произвольным двоичным ядром декодирование методом последовательного исключения сводится к вычислению логарифмических отношений правдоподобия для информационных символов кодов, порождаемых строками ядра поляризации. В данном разделе был предложен алгоритм вычисления таких логарифмических отношений правдоподобия, предусматривающий два режима работы. Переключение между ними осуществляется в зависимости от индекса обрабатываемого символа. Показано, что вычислительная сложность предлагаемого подхода существенно меньше сложности вычисления логарифмических отношений правдоподобия по определению.

### 3.4. Последовательное декодирование полярных кодов с произвольным двоичным ядром

В разделе 3.3 рассматривалась реализация метода последовательного исключения для полярных кодов с произвольным двоичным ядром поляризации. Однако этот алгоритм не способен обеспечить декодирование по максимуму правдоподобия. В данном разделе представлено обобщение последовательного алгоритма декодирования, представленного в разделе 3.2 для случая полярных кодов с ядром Арикана, на случай полярных кодов с произвольным ядром поляризации.

#### 3.4.1. Описание алгоритма декодирования

Обобщение последовательного алгоритма декодирования заключается в построении метода вычисления метрик путей  $\hat{T}(u_0^i, y_0^{n-1})$ , задаваемых выражением (3.2). Для  $l \times l$  ядра  $M$  максимум апостериорной вероятности кодового слова, принадлежащего надкоду исходного полярного кода, вычисляется рекурсивно согласно выражению

$$R(u_0^{ls+t}, y_0^{n-1}) = \max_{u_{ls+t+1}^{l(s+1)-1}} \prod_{j=0}^{l-1} R\left(\theta_M \left[ u_0^{l(s+1)-1}, j \right], y_{j\frac{n}{l}}^{(j+1)\frac{n}{l}-1} \right), \quad (3.9)$$

в основе которого лежит выражение (1.11),  $0 \leq t < l$ . Декодирование полярного кода с произвольным ядром может быть выполнено аналогично декодированию полярных кодов с ядром Арикана, то есть посредством хранения в стеке путей  $u_0^i$ , удлинения на каждой итерации пути с наибольшим значением метрики  $\hat{T}(u_0^i, y_0^{n-1})$ , до тех пор, пока не будет получен путь длины  $n$ .

Однако задача вычисления (3.9) для ядра с  $l > 2$  обладает существенно большей сложностью по сравнению с аналогичной задачей в случае  $2 \times 2$  ядра Арикана. Кроме того, необходимы методы вычисления функции  $\hat{\Omega}(i)$ .

## 3.4.2. Вычисление метрики пути

### 3.4.2.1. Функция $R(u_0^i, y_0^{n-1})$

Пусть  $i = ls + t, 0 \leq t < l$ . Задача поиска последовательности  $u_{ls+t+1}^{l(s+1)-1}$ , при которой достигается максимум (3.9), эквивалентна задаче декодирования в смежном классе  $(l, \kappa = l - t - 1)$  кода  $C_t$ , порожденного последними  $\kappa$  строками матрицы  $M$ . Смежный класс задается вектором  $u_{ls}^{ls+t} M_{0..t} + C_t$ , где  $M_{0..t}$  — матрица, состоящая из первых  $t + 1$  строк матрицы  $M$ . Действительно, задача (3.9) может рассматриваться как задача вычисления вероятности

$$p[u_{ls+t}]_t = \max_{u_{ls+t+1}^{l(s+1)-1}} P(u_{ls}^{l(s+1)-1} | z_0^{l-1}), \quad (3.10)$$

для переходных вероятностей  $P(b|z_j) = \Delta_j R\left((\theta_M[u_0^{ls-1}, j], b), y_j^{\frac{(j+1)n}{t}-1}\right)$ ,  $b \in \{0, 1\}$ ,  $0 \leq j < l$ , где  $\Delta_j$  — некоторых коэффициентов. Максимум апостериорной вероятности в (3.10) может быть найден при мягком декодировании последовательности  $z_0^{l-1}$  в коде  $C_t$ .

Следовательно, сложность последовательного алгоритма декодирования можно оценить в  $O(Ln \log_l(n))$  базовых операций, где базовая операция соответствует поиску наиболее вероятного кодового слова в коде  $C_t$ .

### 3.4.2.2. Функция $\hat{\Omega}(i)$

Задача вычисления вероятностей ошибки  $P_j, j \in \mathcal{F}$ , для битовых подканалов, задаваемых поляризующим преобразованием с произвольным ядром, при условии наличия правильных значений  $u_0^{j-1}$ , остается открытой. В силу чего, мы используем значения вероятностей  $P_j$ , являющиеся результатом статистического моделирования.

В качестве альтернативы может быть использована верхняя граница, построенная на базе методов вычисления спектров, предложенных для многоуровневых кодов в [83], или метод Гауссовской аппроксимации, основанный на разложении выражений ЛОПП символов в ряды Грама-Чарлиера [10]. Применение

первого подхода приводит к крайне неточным оценкам вероятностей  $P_j$  для плохих подканалов, из которых и состоит множество  $\mathcal{F}$ . Недостатком второго подхода является появление расходящихся рядов.

### 3.4.3. Улучшенный алгоритм декодирования

При принятии решений относительно значений входных символов  $u_i$ ,  $sl \leq i < (s+1)l$ , используются только вероятности  $R\left(\left(\theta_M[u_0^{ls-1}, j], b\right), y_j^{\frac{(j+1)l}{t}-1}\right)$ ,  $b \in \{0, 1\}$ , и соответствующее подмножество замороженных символов  $\mathcal{F} \cap \{sl, \dots, sl+l-1\}$ . Заметим, что значения этих вероятностей не зависят от значения  $t = i \bmod l$ . Таким образом, решения относительно значений  $u_{sl}^{(s+1)l-1}$  могут быть приняты совместно, что позволяет существенно снизить сложность декодирования.

Предположим, что на некоторой итерации  $\beta$  алгоритма декодирования, предложенного в разделе 3.4.1, выбран путь  $u_0^{sl-1}$  как наиболее вероятный. Определим множество путей длины  $(s+1)l$ , которые могут быть получены из пути  $u_0^{sl-1}$

$$V = \left\{ u_0^{(s+1)l-1} \mid u_{sl}^{(s+1)l-1} \in \{0, 1\}^l, u_{sl, \mathcal{F}}^{(s+1)l-1} = \mathbf{0} \right\}.$$

Напомним, что последовательный алгоритм декодирования предполагает удаление путей с наименьшими метриками из стека во избежание его переполнения. Общее количество путей в стеке не должно превосходить  $\Theta$ , а число путей одной длины не должно превосходить  $L$ . Для упрощения изложения, предположим, что все пути из множества  $V$  добавляются в стек, после чего поочередно удаляются наименее вероятные пути до тех пор, пока стек остается переполненным. Пусть  $\Phi \subset V$  — множество путей, которые не были удалены.

Из выражения (3.2) видно, что метрики  $\hat{T}(u_0^{(s+1)l-1}, y_0^{n-1})$  имеют общий множитель  $\hat{\Omega}((s+1)l-1)$ , то есть  $\Phi$  состоит из  $\phi = |\Phi|$  путей  $u_0^{(s+1)l-1}$  с наибольшими вероятностями  $R(u_0^{(s+1)l-1}, y_0^{n-1})$ . Следовательно, эти пути соответствуют  $\phi$  наиболее вероятным кодовым словам  $u_{sl}^{(s+1)l-1} M$  кода  $\Upsilon_s$  с парамет-



рами  $(l, \lambda_s)$ , порожденного строками матрицы  $M$  с индексами  $i: sl + i \in \Xi_s$ , где  $\Xi_s = \{sl, sl + 1, \dots, (s + 1)l - 1\} \setminus \mathcal{F}$  — множество индексов информационных символов в  $s$ -ом блоке входной последовательности и  $\lambda_s = |\Xi_s|$ . Такие кодовые слова могут быть найдены с помощью списочного алгоритма мягкого декодирования для двоичного линейного кода  $\Upsilon_s$ .

Пусть  $\hat{t}_j$  —  $j$ -ая наибольшая метрика пути, хранимая в стеке на итерации  $\beta$ ,  $0 \leq j < \Theta$ . Если стек содержит  $\tilde{\Theta} < \Theta$  путей, то будем полагать, что  $\hat{t}_j = 0$ ,  $\tilde{\Theta} \leq j < \Theta$ . Пусть  $Q_i$  — множество путей длины  $i$ , хранимых в стеке. Пусть  $\tilde{t}_j$  —  $j$ -ая наибольшая метрика пути из множества  $Q_{(s+1)l}$ . Пусть  $\tilde{t}_j = 0$ ,  $\tilde{L} \leq j < L$ , где  $\tilde{L} = |Q_{(s+1)l}|$ .  $\Phi$  содержит все пути из  $V$ , оставшиеся в стеке после запуска процедуры удаления избыточных путей, то есть

$$u_0^{(s+1)l-1} \in V : \hat{T}(u_0^{(s+1)l-1}, y_0^{n-1}) \geq \max(\hat{t}_{\Theta-\phi}, \tilde{t}_{L-\phi}). \quad (3.11)$$

Заметим, что если  $L - \tilde{L} \geq 2^\lambda$  и  $\Theta - \tilde{\Theta} \geq 2^{\lambda_s}$ , то выполняется полный перебор по всем кодовым словам  $w_0^{l-1} \in \Upsilon_s$ , и для каждого  $w_0^{l-1}$  соответствующий путь вместе со своей метрикой добавляется в стек. Множество путей  $\Phi$ , задаваемых выражением (3.11), оказывается чрезмерно избыточным. Эффективность алгоритма декодирования может быть повышена путей замены  $\Phi$  на

$$\hat{\Phi} = \left\{ u_0^{(s+1)l-1} \in \Phi \mid \hat{T}(u_0^{(s+1)l-1}, y_0^{n-1}) \geq e^{-\alpha} \rho \right\}, \quad (3.12)$$

где  $\rho = \hat{T}((u_0^{sl}, w_0^{l-1}), y_0^{n-1})$ , и  $w_0^{l-1}M$  — наиболее вероятное кодовое слово кода  $\Upsilon_s$ . Параметр  $\alpha$  определяет размер получаемого списка кодовых слов. Необходимо заметить, что при достаточно больших значениях  $\alpha$  может потребоваться декодирование в коде  $\Upsilon_s$  за границей Джонсона [25], что может привести к экспоненциальной зависимости размера списка от величины  $l$ . Для получения практически применимого алгоритма необходимо установить ограничение на наибольший возможный размер списка  $\hat{\Phi}$ .

Улучшенный декодер работает следующим образом. На каждой итерации для удлинения выбирается путь  $u_0^{sl-1}$  с наибольшей метрикой. Выполняется списочное декодирование в коде  $\Upsilon_s$ , полученное множество путей (3.12) помещается

в стек вместе со своими метриками. Заметим, что одна итерация соответствует удлинению пути на  $l$  символов. Входными данными для списочного декодера являются вероятности  $R\left(\left(\theta_M[u_0^{ls-1}, j], b\right), y_{j\frac{n}{l}}^{(j+1)\frac{n}{l}-1}\right)$ ,  $b \in \{0, 1\}$ . Эти вероятности задаются (3.9), при этом они рекурсивно вычисляются декодерами кодов, порожденных последними строками матрицы  $M$ . Метрика пути определена как

$$\hat{T}(u_0^{(s+1)l-1}, y_0^{n-1}) = R(u_0^{(s+1)l-1}, y_0^{n-1})\tilde{\Omega}(s+1),$$

где

$$\hat{\Omega}(s) = \prod_{\sigma=s+1}^{n/l-1} (1 - \Gamma_\sigma),$$

и  $\Gamma_\sigma$  — вероятность появления ненулевого значения хотя бы на одной из позиций  $j \in \{\sigma l, \dots, (\sigma+1)l-1\} \cap \mathcal{F}$  наиболее вероятного пути с началом  $u_0^{\sigma l-1}$ . Напомним, что во избежании переполнения стека могут удаляться наименее вероятные пути. Вероятности  $\Gamma_\sigma$  могут быть получены посредством статистического моделирования. Декодирование прекращается при появлении на вершине стека пути длины  $n$ .

#### 3.4.4. Реализация

Операция кодирования полярных кодов длины  $n = l^m$  может рассматриваться как  $m$ -слойное преобразование, каждый слой соответствует применению линейного преобразования, задаваемого матрицей  $M$ , к  $l^{m-1}$  блокам данных, полученным на предыдущем слое. В случае декодирования  $(m-1)$ -ый слой (последний слой) соответствует восстановлению значений входных символов  $u_i$ , при котором используется списочный декодер, как описано в разделе 3.4.3. На слоях  $0, \dots, m-2$  (промежуточные слои) необходимо выполнять поиск наиболее вероятного кодового слова, задаваемого (3.10), для каждого блока данных.

### 3.4.4.1. Промежуточные слои

При поиске решения (3.10) может быть использован любой алгоритм мягкого декодирования для смежного класса  $u_{l_s}^{l_s+t} M_{0..t} + C_t$ . Рассмотрим случай применения алгоритма Voh-and-Match, предложенного в [80]. Для кода размерности  $\kappa$ , этот алгоритм с параметрами  $(\kappa + \psi, \gamma)$  способен исправить до  $2\gamma$  ошибок на позициях из множества  $\Lambda^{(t)} \cup \Psi^{(t)}$ , где  $\Lambda^{(t)}$  — наиболее надежная информационная совокупность и  $\Psi^{(t)}$  — множество, состоящее из  $\psi$  наиболее надежных позиций за исключением позиций из множества  $\Lambda^{(t)}$ . Параметр  $\psi$  позволяет сбалансировать вычислительную сложность и объем потребляемой памяти.

Для заданных входных вероятностей  $\omega[b]_j = P(b + u_{l_s}^{l_s+t} M_{0..t,j} | z_j)$  вычисляется вектор ЛОПП  $\chi_0^{l-1}$ :

$$\chi_j = (-1)^{u_{l_s}^{l_s+t} M_{0..t,j}} \log \frac{\omega[1]_j}{\omega[0]_j},$$

где  $M_{0..t,j}$  — вектор, состоящий из первых  $t + 1$  элементов  $j$ -го столбца матрицы  $M$ . Для того, чтобы найти наиболее вероятное кодовое слово  $v_{t+1}^{l-1} M_{t+1..l-1} \in C_t$  строятся множества  $\Lambda^{(t)}$  и  $\Psi^{(t)}$ . На первом шаге алгоритма выполняется сортировка вектора надежности  $(|\chi_0|, \dots, |\chi_{l-1}|)$ . Можно показать, что перестановка, получаемая в результате выполнения этого шага не зависит от  $t$ , то есть для каждого заданного  $s$  этот шаг достаточно выполнить только один раз. Заметим, что  $\Lambda^{(0)} \supset \Lambda^{(1)} \supset \dots \supset \Lambda^{(l-2)}$ . Для снижения сложности мы предлагаем совместно построить информационные совокупности  $\Lambda^{(t)}$ . В предположении, что  $C_{l-2}$  является кодом с повторением, получаем  $\Lambda^{(l-2)} = \{j_{l-2}\} : j_{l-2} = \arg \max_j |\chi_j|$ . Для заданной информационной совокупности  $\Lambda^{(t)}$  кода  $C_t$ , можно построить информационную совокупность  $\Lambda^{(t-1)} = \Lambda^{(t)} \cup \{j_{t-1}\}$  для кода  $C_{t-1}$ . Вектор  $M_{t..l-1, j_{t-1}}$  должен быть линейно независим от векторов  $M_{t..l-1, i}, i \in \Lambda^{(t)}$ , при этом выбирается наибольшее возможное значение  $|\chi_{j_{t-1}}|$ . Множество  $\Psi^{(t)}$  может быть построено схожим образом. Для получаемой на выходе алгоритма последовательности  $v_{t+1}^{l-1} M_{t+1..l-1}$  следует вычислить вероятность  $p[v_t]_t = \prod_{j=0}^{l-1} \omega[(v_0^{l-1} M)_j]_j$ . Даже если последовательность  $v_{t+1}^{l-1} M_{t+1..l-1}$  не является наиболее вероятным ре-

шением задачи декодирования в силу субоптимальности используемого алгоритма, результат работы предлагаемого алгоритма декодирования может оказаться правильным, поскольку значения вероятностей  $p[v_t]_t$  обычно близки к своим истинным значениям.

Напомним, что для любого  $t$  необходимо вычислить вероятности  $p[0]_t$  и  $p[1]_t$ , для чего выполняется поиск двух наиболее вероятных кодовых слов  $c[0]_0^{l-1}$  и  $c[1]_0^{l-1}$  в смежных классах  $(v_0^{t-1}, 0)M_{0..t} + C_t$  и  $(v_0^{t-1}, 1)M_{0..t} + C_t$ , соответственно. При этом известно, что наиболее вероятное кодовое слово  $c_0^{l-1}$  в смежном классе  $v_0^{t-1}M_{0..t-1} + C_{t-1}$  уже было вычислено на шаге  $t - 1$ . Очевидно, что кодовое слово  $c[0]_0^{l-1}$  или  $c[1]_0^{l-1}$  равно  $c_0^{l-1}$ , таким образом, на  $t$ -ом шаге требуется только один раз вызвать декодер для кода  $C_t$ .

Сложность вычисления  $R(u_0^{sl+t}, y_0^{n-1})$ ,  $u_{ls+t} \in \{0, 1\}$ ,  $0 \leq t < l$ , на промежуточных слоях декодера складывается из сложности сортировки вектора надежности  $O(l \log l)$ , сложности построения информационных совокупностей  $O(l^3)$  и сложности поиска наиболее вероятных кодовых слов  $\sum_{t=0}^{l-2} N(l, l-t-1, \psi, \gamma)$ , где  $N(l, \kappa, \psi, \gamma)$  — сложность алгоритма Vox-and-Match с параметрами  $(\kappa + \psi, \gamma)$  при декодировании в коде  $(l, \kappa)$ , исключая операции сортировки и построения информационной совокупности. Число промежуточных слоев равно  $m - 1$ , таким образом, сложность выполнения операций на данных слоях составляет  $(m - 1)l^{m-1} \left( O(l \log l + l^3) + \sum_{t=0}^{l-2} N(l, l-t-1, \psi, \gamma) \right)$ .

#### 3.4.4.2. Последний слой

На последнем слое необходимо построить список кодовых слов, удовлетворяющих условию (3.12), что может быть реализовано с помощью алгоритма декодирования [32], основанного на порядковых статистиках, или алгоритма [52], являющегося обобщением алгоритма [80] на случай списочного декодирования.

Заметим, что при декодировании полностью замороженных блоков входных символов нет необходимости в выполнении каких либо вычислений как на по-

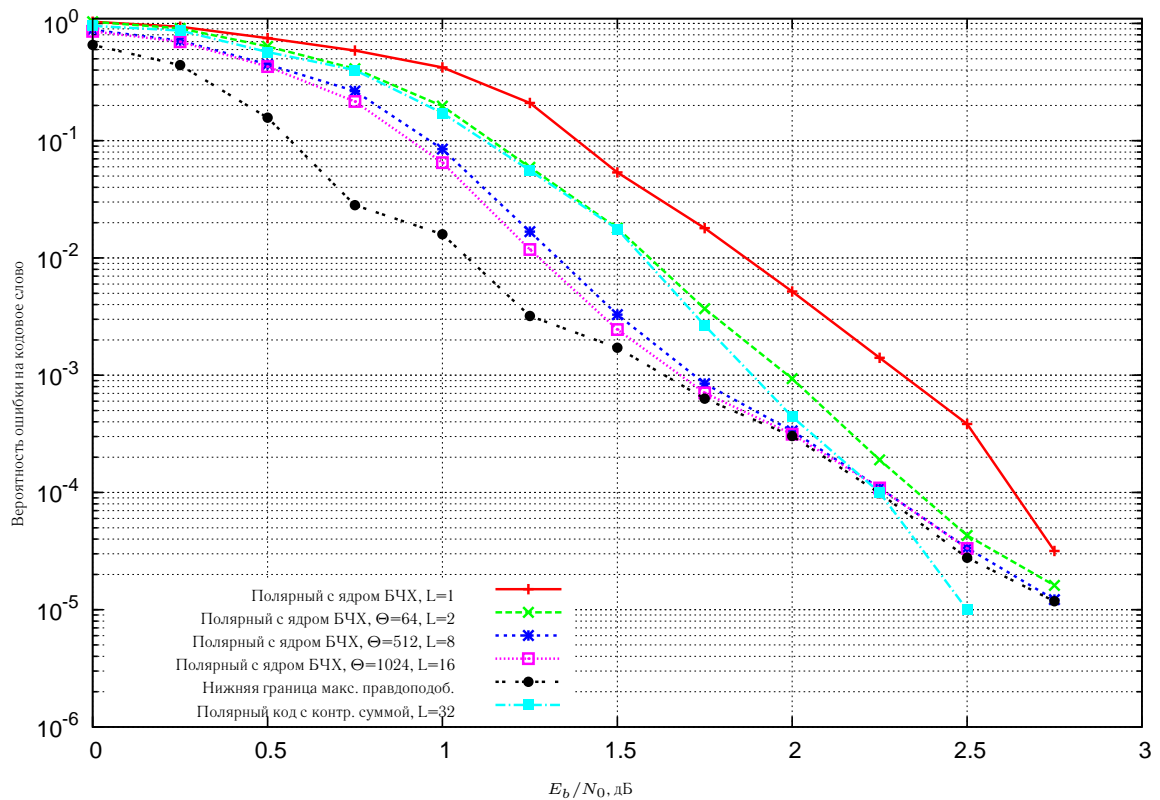


Рис. 3.10. Вероятность ошибки декодирования (1024, 512) полярных кодов с  $32 \times 32$  ядром БЧХ следнем слое, так и на промежуточных слоях.

### 3.4.5. Численные результаты

На Рис. 3.10 представлен график зависимости вероятности ошибки декодирования (1024, 512, 16) полярного кода<sup>1</sup> с  $32 \times 32$  ядром БЧХ, структура которого описана в разделе 1.1.2, и случая передачи двоичного образа по аддитивному Гауссовскому каналу. Для сравнения, представлены результаты для полярного кода с ядром Арикана и с контрольной суммой CRC, для декодирования которого применялся списочный алгоритм последовательного исключения, описанный в разделе 1.3.2. Из графика видно, что при достаточно большом  $L$  предлагаемый алгоритм обеспечивает вероятность ошибки декодирования, близкую к вероятности ошибки декодирования по максимуму правдоподобия. На малых отно-

<sup>1</sup> Данный полярный код был построен с использованием метода, предложенного в разделе 2.3, и имеет коды  $\Upsilon_s, 0 \leq s < n/l$ , с размерностями 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 5, 7, 11, 13, 16, 21, 21, 24, 26, 26, 28, 29, 31, 31, 31, 31, 31, 31, 32, 32, 32, 32.

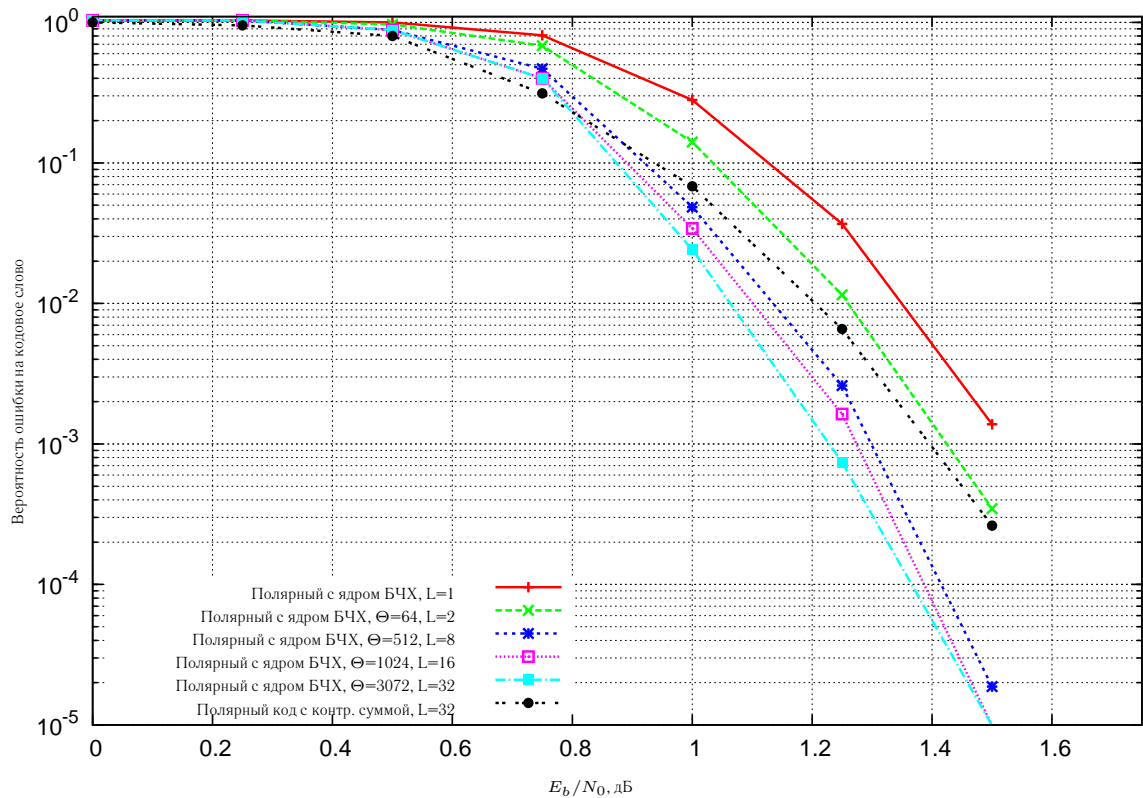


Рис. 3.11. Вероятность ошибки декодирования (4096, 2048) полярных кодов с  $64 \times 64$  ядром БЧХ в условиях сигнал/шум полярные коды с ядром БЧХ превосходят полярные коды Арикана с контрольной суммой CRC. Однако, в области больших отношений сигнал/шум корректирующая способность полярных кодов с ядром БЧХ ограничивается малым минимальным расстоянием. На Рис. 3.11 приведен аналогичный график для случая (4096, 2048, 32) полярного кода с ядром БЧХ и  $l = 64$ . Несмотря на то, что минимальное расстояние кода остается небольшим, скорость убывания вероятности ошибки не уменьшается по крайней мере до 1.5 дБ.

Рис. 3.12–3.13 характеризуют среднее число итераций, выполняемых предлагаемым алгоритмом при декодировании рассматриваемых кодов. Учитываются только итерации, соответствующие блокам входных символов, содержащих информационные символы. Из графиков видно, что при достаточно больших отношениях сигнал/шум, при которых вероятность ошибки декодирования меньше  $10^{-3}$ , предлагаемому алгоритму требуется в среднем число итераций, равное числу блоков, содержащих информационные символы, которое ограничено сверху

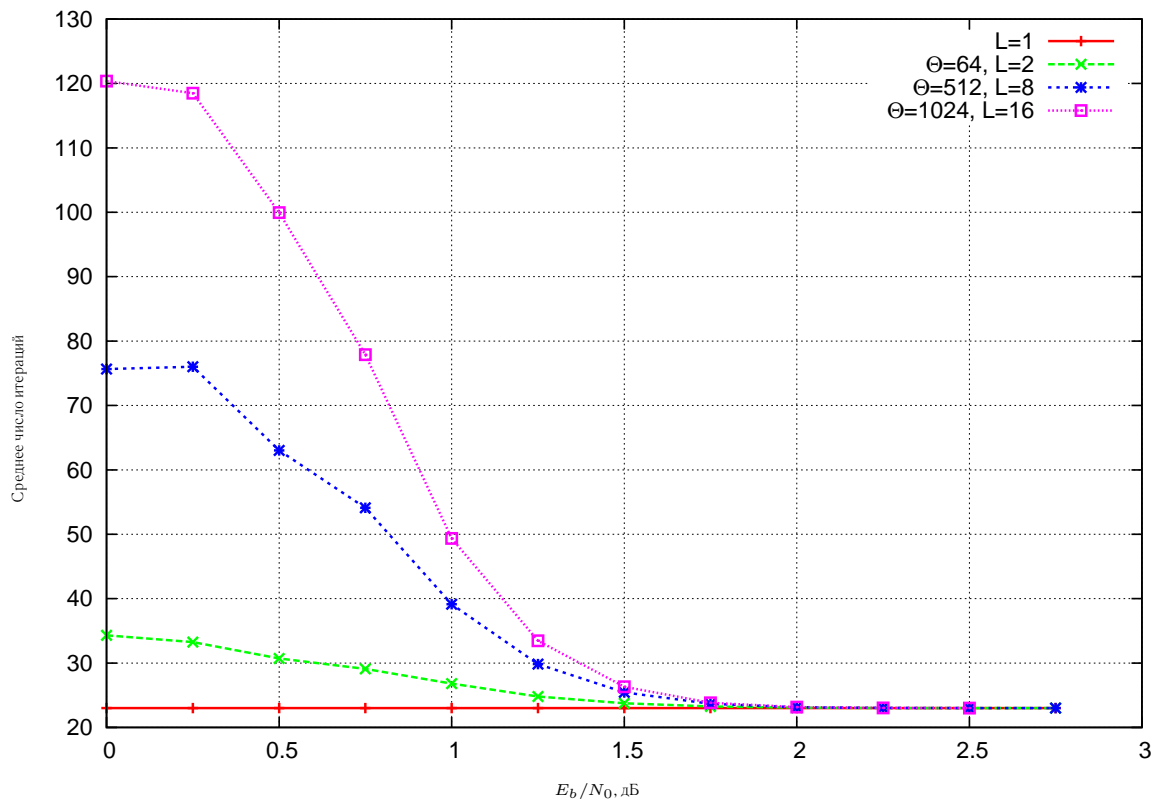


Рис. 3.12. Средняя сложность декодирования (1024, 512) полярного кода с  $32 \times 32$  ядром БЧХ

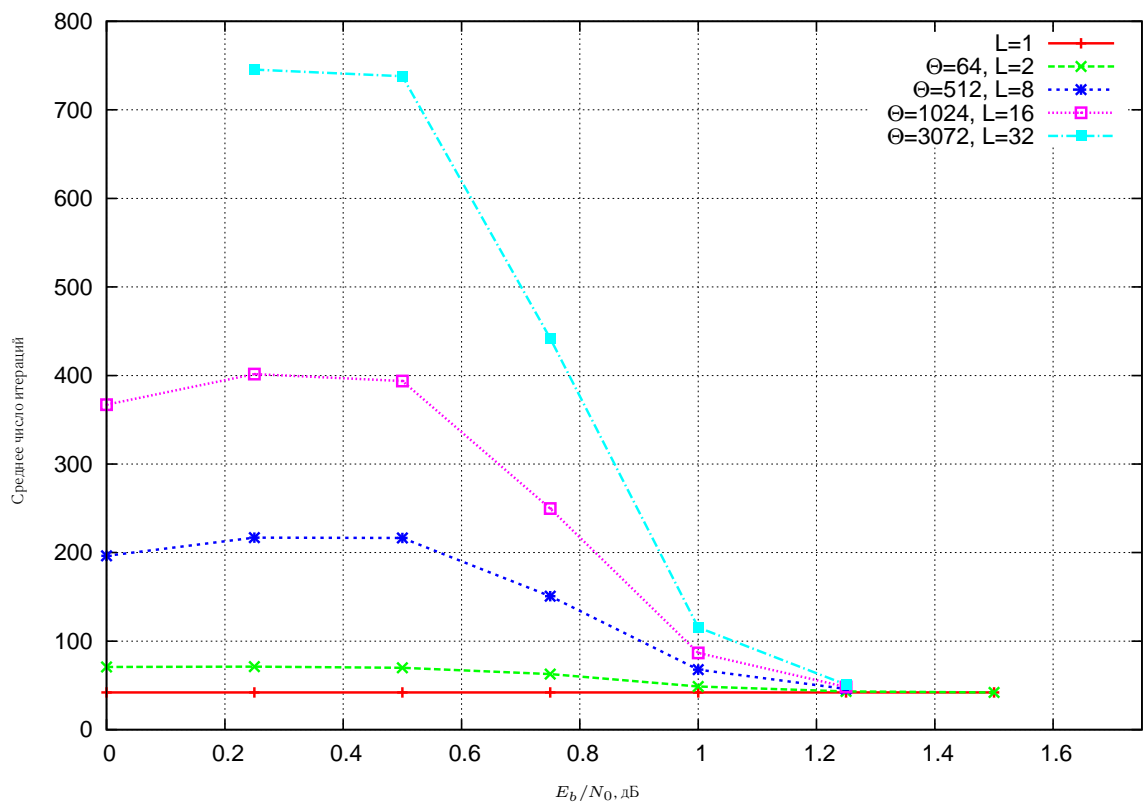


Рис. 3.13. Средняя сложность декодирования (4096, 2048) полярного кода с  $64 \times 64$  ядром БЧХ

$n/l$ .

### 3.4.6. Выводы

В данном разделе предложен новый алгоритм декодирования полярных кодов с произвольным двоичным ядром. Этот алгоритм является обобщением алгоритма последовательного декодирования, предложенного в разделе 3.2 для случая полярных кодов с ядром Арикана. Алгоритм предполагает использование методов декодирования почти по максимуму правдоподобия для кодов, порожденных строками ядра. Численные результаты показывают, что предлагаемый подход позволяет выполнять декодирование полярных кодов с ядром БЧХ почти по максимуму правдоподобия. Показано, что такие коды превосходят по корректирующей способности полярные коды Арикана с контрольной суммой CRC по крайней мере при малых отношениях сигнал/шум.

## 3.5. Последовательное декодирование кодов Рида-Соломона

$(n = 2^m, k, 2^m - k + 1)$  расширенный код Рида-Соломона можно определить как множество векторов  $(c_0, \dots, c_{n-1})$ , где  $c_i = f(a_i)$ ,  $a_i$  — различные элементы поля  $\mathbb{F}_{2^m}$ , и  $\deg f(x) < k$ .

Мы предлагаем применить методы, разработанные для полярных кодов, для мягкого декодирования кодов Рида-Соломона над полем  $\mathbb{F}_{2^m}$ . Действительно, для  $(n, k)$  расширенного кода Рида-Соломона может быть построена система уравнений 2.1, что позволяет использовать при его декодировании метод последовательного исключения и его списочные/стековые аналоги.

В [9] было показано, что если элементы  $a_i$  упорядочены согласно своему представлению в виде двоичных векторов, то множество динамически замороженных символов  $(n, k)$  расширенного кода Рида-Соломона равно  $\mathcal{F} = \{0, \dots, 2^m - 1\} \setminus \{2^m - 1 - r_m(i) \mid 0 \leq i < k\}$ , где  $r_m(i)$  — целое число, полученное из  $i$  в результате перестановки  $m$  битов в обратном порядке. Заметим, что



множество замороженных символов не зависит от используемого базиса поля  $\mathbb{F}_{2^m}$ .

### 3.5.1. Алгоритм последовательного исключения

Как показано в разделе 1.3.1.1, задача  $i$ -го шага алгоритма последовательного исключения состоит в вычислении вероятностей  $P(u_0^i | y_0^{n-1})$  для заданной последовательности  $u_0^{i-1}$  и различных значений  $u_i$ , где  $0 \leq i < n$ . Этот метод может быть обобщен на случай полярных кодов с ядром Арикана над полем  $\mathbb{F}_{2^m}$ , при этом выражение (1.11) преобразуется в

$$P(u_0^{2i}, y_0^{n-1}) = \sum_{u_{2i+1} \in \mathbb{F}_{2^m}^{n-i-1}} P(u_{0,E}^{2i+1} \oplus u_{0,O}^{2i+1} | y_0^{n/2-1}) P(u_{0,O}^{2i+1} | y_{n/2}^{n-1}), \quad (3.13)$$

$$P(u_0^{2i+1}, y_0^{n-1}) = P(u_{0,E}^{2i+1} \oplus u_{0,O}^{2i+1} | y_0^{n/2-1}) P(u_{0,O}^{2i+1} | y_{n/2}^{n-1}). \quad (3.14)$$

Выражение (3.13) может быть эффективно вычислено с помощью метода, предложенного в [23]. Для этого распределения вероятностей представляются в виде массивов  $\chi^{(0)}$ :  $\chi_r^{(0)} = P(u_{0,E}^{2i-1} \oplus u_{0,O}^{2i-1}, r | y_0^{n/2-1})$  и  $\chi^{(1)}$ :  $\chi_r^{(1)} = P(u_{0,O}^{2i-1}, r | y_{n/2}^{n-1})$ ,  $r \in \mathbb{F}_{2^m}$ . После чего вычисляется

$$\chi = \mathcal{H}^{-1} \left( \mathcal{H}(\chi^{(0)}) \odot \mathcal{H}(\chi^{(1)}) \right), \quad (3.15)$$

где  $\chi_r = P(u_0^{2i-1}, r | y_0^{n-1})$ ,  $\odot$  служит для обозначения покомпонентного произведения двух векторов. Здесь  $f = \mathcal{H}(\chi)$  — вектор, получаемый из вектора  $\chi$  в результате применения преобразования Адамара, задаваемого выражением

$$f_r = \sum_{j \in \mathbb{F}_{2^m}} (-1)^{r \boxplus j} \chi_j, \quad r \in \mathbb{F}_{2^m}, \quad (3.16)$$

где  $r \boxplus j$  — покомпонентное произведение двух элементов поля  $\mathbb{F}_{2^m}$ , представленных в виде двоичных векторов.

### 3.5.2. Предлагаемый алгоритм

Мы предлагаем обобщить алгоритм, описанный в разделе 3.2, на случай полярных кодов над  $\mathbb{F}_{2^m}$  и применить полученный алгоритм для декодирования

( $n = 2^m, k, n-k+1$ ) расширенного примитивного кода Рида-Соломона. Переход от двоичных полярных кодов к полярным кодам над  $\mathbb{F}_{2^m}$  не ведет к значительным изменениям в алгоритме. Если  $i \in \mathcal{F}$ , то путь  $u_0^{i-1}$  удлиняется на символ, значение которого вычисляется согласно выражению (2.1), иначе путь копируется для получения  $2^m$  путей  $u_0^i = (u_0^{i-1}, r)$ ,  $r \in \mathbb{F}_{2^m}$ . При этом необходимо вычислить оценки для этих  $2^m$  удлиненных путей

$$\hat{T}(u_0^i, y_0^{n-1}) = R_{2^m}(u_0^i, y_0^{n-1})\hat{\Omega}(i), \quad (3.17)$$

где

$$R_{2^m}(u_0^i, y_0^{n-1}) = \max_{u_{i+1}^{n-1} \in \mathbb{F}_{2^m}^{n-i-1}} P(u_0^{n-1} | y_0^{n-1}), \quad (3.18)$$

$$\hat{\Omega}(i) = \prod_{j \in \mathcal{F}, j > i} (1 - P_j^{(2^m)}), \quad (3.19)$$

где  $P_j^{(2^m)}$  — вероятность ошибки в  $j$ -ом подканале при условии того, что даны значения символов  $u_{j'} \in \mathbb{F}_{2^m}$ ,  $j' < j$ . Функция  $\hat{\Omega}(i)$  равна вероятности того, что при декодировании  $u_{i+1}^{n-1}$  без учета наличия замороженных символов, значения этих замороженных символов будут удовлетворять условиям динамической заморозки. Значение  $\hat{\Omega}(i)$  зависит только от  $n$ ,  $\mathcal{F}$  (то есть рассматриваемого кода), свойств канала передачи данных и фазы  $i$ . Необходимо разработать методы, позволяющие оценить значение  $\hat{\Omega}(i)$ .

Как и в двоичном случае, вычисление вероятностей  $R_{2^m}(u_0^i, y_0^{n-1})$  для кода длины  $n$  сводится к вычислению вероятностей для кода длины  $n/2$ , то есть

$$R_{2^m}(u_0^{2i}, y_0^{n-1}) = \max_{u_{2i+1}^{n-1} \in \mathbb{F}_{2^m}^{n-2i-1}} R_{2^m}(u_{0,E}^{2i+1} \oplus u_{0,O}^{2i+1}, y_0^{n/2-1}) R_{2^m}(u_{0,O}^{2i+1}, y_{n/2}^{n-1}), \quad (3.20)$$

$$R_{2^m}(u_0^{2i+1}, y_0^{n-1}) = R_{2^m}(u_{0,E}^{2i+1} \oplus u_{0,O}^{2i+1}, y_0^{n/2-1}) R_{2^m}(u_{0,O}^{2i+1}, y_{n/2}^{n-1}) \quad (3.21)$$

с начальным условием  $R_{2^m}(b, y_j) = P(b|y_j)$ ,  $b \in \mathbb{F}_{2^m}$ .

Вычисления могут быть упрощены. Если взять логарифмы от обеих частей выражений (3.17)–(3.21), то при декодировании потребуются только операции сложения и сравнения. Кроме того, достаточно хранить несколько наибольших

значений вероятностей  $R_{2^m}(u_0^i, y_0^{n-1})$  для промежуточных символов, такой подход был предложен в [49] для снижения сложности декодирования кодов МППЧ над  $\mathbb{F}_{2^m}$ .

Сложность предлагаемого метода не превосходит  $O(Ln \log n)$  базовых операций, где одна базовая операция соответствует вычислению (3.20) или (3.21). Заметим, что  $R_{2^m}(u_0^i, y_0^{n-1})$  должны быть вычислены для  $2^m$  различных значений  $u_i$ . Поскольку максимизация в выражении (3.20) выполняется над полем  $\mathbb{F}_{2^m}$  и  $n = 2^m$ , получаем в худшем случае сложность одной базовой операции  $O(n^2)$ . Таким образом, сложность декодирования расширенных кодов Рида-Соломона можно оценить как  $O(Ln^3 \log n)$  операций над вещественными числами. Необходимо отметить, что при увеличении длины кода  $n$  требуемый размер списка  $L$  растет экспоненциально.

### 3.5.3. Вычисление функции $\hat{\Omega}(i)$

#### 3.5.3.1. Гауссовская аппроксимация

Для получения значения  $\hat{\Omega}(t)$  необходимо вычислить вероятности  $P_\gamma^{(2^m)}$ . В случае двоичных полярных кодов и аддитивного Гауссовского канала для этой цели использовалась Гауссовская аппроксимация, описанная в разделе 1.3.1.2.

Для кодов над  $\mathbb{F}_{2^m}$  может быть использовано обобщение этого метода, описанное в [48] для случая кодов МППЧ. Рассмотрим случай передачи нулевого кодового слова по аддитивному Гауссовскому каналу с  $2^m$ -ичным входом, так что ЛОПП  $l_r^{(\gamma)} = \log \frac{P(u_0^{\gamma-1}, 0 | y_0^{n-1})}{P(u_0^{\gamma-1}, r | y_0^{n-1})}$ ,  $r \in \mathbb{F}_{2^m} \setminus \{0\}$ , могут рассматриваться как зависимые Гауссовские случайные величины с математическими ожиданиями  $\rho_r^{(\gamma)}$  и  $(2^m - 1) \times (2^m - 1)$  ковариационной матрицей  $\Sigma^{(\gamma)}$ :

$$\Sigma_{i,j}^{(\gamma)} = \rho_i^{(\gamma)} + \rho_j^{(\gamma)} - \rho_{i \oplus j}^{(\gamma)}, \quad i, j \in \mathbb{F}_{2^m} \setminus \{0\}, \quad (3.22)$$

где  $i \oplus j$  — число, получаемое в результате применения операции “исключающее или” к соответствующим битам чисел  $i$  и  $j$ .

Пусть  $\rho_r^{(0,\gamma)}$  и  $\rho_r^{(1,\gamma)}$  являются математическими ожиданиями ЛОПП  $l_r^{(0,\gamma)} = \log \frac{P(u_{0,E}^{2\gamma-1} \oplus u_{0,O}^{2\gamma-1}, 0 | y_0^{n/2-1})}{P(u_{0,E}^{2\gamma-1} \oplus u_{0,O}^{2\gamma-1}, r | y_0^{n/2-1})}$  и  $l_r^{(1,\gamma)} = \log \frac{P(u_{0,O}^{2\gamma-1}, 0 | y_0^{n/2-1})}{P(u_{0,O}^{2\gamma-1}, r | y_0^{n/2-1})}$ , соответственно. Рассмотрим задачу вычисления векторов математических ожиданий  $\rho^{(2\gamma)}$  и  $\rho^{(2\gamma+1)}$  для заданных  $\rho^{(0,\gamma)}$  и  $\rho^{(1,\gamma)}$ . Как было показано в [48], они могут быть вычислены как

$$\rho^{(2\gamma)} = \Phi_{2^m-1}^{-1}(\Phi_{2^m-1}(\rho^{(0,\gamma)}) \odot \Phi_{2^m-1}(\rho^{(1,\gamma)})) \quad (3.23)$$

$$\rho^{(2\gamma+1)} = \rho^{(0,\gamma)} + \rho^{(1,\gamma)}, \quad (3.24)$$

где  $a \odot b$  обозначает покомпонентное произведение векторов  $a$  и  $b$ ,  $\Phi_{2^m-1}(\rho)$  — вектор математических ожиданий значений элементов преобразования Адамара, вычисленных для  $l$ , где  $\rho_r = E[l_r]$ , то есть  $(\Phi_{2^m-1}(\rho))_r = E[f_r]$  при

$$f_r = \frac{\sum_{i \in \mathbb{F}_{2^m}} (-1)^{r \boxplus i} \exp(-l_i)}{\sum_{i \in \mathbb{F}_{2^m}} \exp(-l_i)}, \quad r \in \mathbb{F}_{2^m} \setminus \{0\},$$

где  $r \boxplus i$  соответствует побитовому произведению чисел  $r$  и  $i$ . Можно вычислить  $E[f_r] = E[\tanh(z_r/2)]$ , где

$$z_r = \log \frac{\sum_{i \in \Psi(r)} \exp(-l_i)}{\sum_{i \in \Psi(r)} \exp(-l_{i \oplus h(r)})},$$

где  $\Psi(r)$  — линейное подпространство  $\mathbb{F}_{2^m}$ , и  $h^{(r)} : r \boxplus h^{(r)} = 1$ . Рекурсивные выражения для вычисления математического ожидания величины  $z_r$  приведены в [48].

В итоге получаем, что

$$P_\gamma^{(2^m)} = 1 - \prod_{r=1}^{2^m-1} \left(1 - P_{\gamma,r}^{(2^m)}\right),$$

где  $P_{\gamma,r}^{(2^m)} = Q\left(\sqrt{\tilde{\rho}_r^{(\gamma)}/2}\right)$ , и  $\tilde{\rho}^{(\gamma)} = (\Sigma^{(\gamma)})^{-1/2} \rho^{(\gamma)}$ ,  $r \in \mathbb{F}_{2^m}$ . Необходимость умножения вектора  $\rho^{(\gamma)}$  на матрицу  $(\Sigma^{(\gamma)})^{-1/2}$  связана с тем, что случайные величины  $l_r^{(\gamma)}$  являются статистически зависимыми. Значения  $P_\gamma^{(2^m)}$  подставляются в (3.19) для получения  $\hat{\Omega}(t)$ .

### 3.5.3.2. Упрощенный метод

Вычисление (3.23) требует численного решения системы нелинейных уравнений. Даже для  $m = 4$  это приводит к возникновению больших погрешностей. В качестве альтернативы этому методу мы предлагаем упрощенный подход, в котором используется аппроксимация значения суммы значением наибольшего слагаемого:  $P(u_0^{2\gamma}|y_0^{n-1}) = \sum_{u_{2\gamma+1} \in \mathbb{F}_{2^m}} P(u_{0,E}^{2\gamma+1} \oplus u_{0,O}^{2\gamma+1}|y_0^{n/2-1}) P(u_{0,O}^{2\gamma+1}|y_{n/2}^{n-1}) \approx \max_{u_{2\gamma+1} \in \mathbb{F}_{2^m}} P(u_{0,E}^{2\gamma+1} \oplus u_{0,O}^{2\gamma+1}|y_0^{n/2-1}) P(u_{0,O}^{2\gamma+1}|y_{n/2}^{n-1})$ , отсюда получается следующая аппроксимация:  $l_r^{(2\gamma)} \approx \lambda_r^{(2\gamma)} - \lambda_0^{(2\gamma)}$ , где

$$\lambda_r^{(2\gamma)} = \min_{i \in \mathbb{F}_{2^m}} (l_i^{(0,\gamma)} + l_{i \oplus r}^{(1,\gamma)}), r \in \mathbb{F}_{2^m},$$

тогда  $\rho_r^{(2\gamma)} \approx \xi_r^{(2\gamma)} - \xi_0^{(2\gamma)}$ , где  $\xi_r^{(2\gamma)} = E[\lambda_r^{(2\gamma)}]$ .

Для заданного ненулевого  $r$  определим случайную величину  $\tilde{X}_i = l_i^{(0,\gamma)} + l_{i \oplus r}^{(1,\gamma)}$ ,  $i \in \mathbb{F}_{2^m}$ , ее математическое ожидание  $\mu_i = E[\tilde{X}_i] = \rho_i^{(0,\gamma)} + \rho_{i \oplus r}^{(1,\gamma)}$ . Заметим, что эти случайные величины зависимы и распределены по-разному (ЗРР). Мы по-прежнему предполагаем, что  $l_i^{(0,\gamma)}$  и  $l_i^{(1,\gamma)}$  являются Гауссовскими случайными величинами, а тогда  $\tilde{X}_i$  также являются Гауссовскими случайными величинами. В [66] был предложен метод оценки математического ожидания максимума ЗРР Гауссовских случайных величин. Воспользуемся этим методом при оценке  $\xi_r^{(2\gamma)} = E[\min_i \tilde{X}_i]$ . Идея метода в [66] состоит в рассмотрении новых случайных величин таких, что математическое ожидание максимума из них может быть легко вычислено, при этом это математическое ожидание является верхней/нижней границей для математического ожидания максимума из исходных ЗРР случайных величин. В основе метода лежит теорема из [81], которая была изложена в [66] следующим образом. Рассмотрим центрированные ЗРР Гауссовские случайные величины  $\bar{W}_i, \bar{X}_i$  и  $\bar{Y}_i, i \in \mathbb{F}_{2^m}$ , где  $\bar{X}_i = \tilde{X}_i - \mu_i$  такие, что для всех  $i, j$

$$D[\bar{W}_i - \bar{W}_j] \leq D[\bar{X}_i - \bar{X}_j] \leq D[\bar{Y}_i - \bar{Y}_j]. \quad (3.25)$$

Здесь  $D[A]$  — дисперсия случайной величины  $A$ . Тогда

$$E[\max_i \bar{W}_i] \leq E[\max_i \bar{X}_i] \leq E[\max_i \bar{Y}_i],$$

где  $\tilde{W}_i = \overline{W}_i + \mu_i$  и  $\tilde{Y}_i = \overline{Y}_i + \mu_i$  для любых  $\mu_i$ .

Соответственно, в случае поиска минимума мы получаем, что

$$E[\min_i \tilde{W}_i] \geq E[\min_i \tilde{X}_i] \geq E[\min_i \tilde{Y}_i], \quad (3.26)$$

Мы не можем вычислить значение  $E[\min_i \tilde{X}_i]$ , однако, мы можем получить его оценку, используя его верхнюю границу  $E[\min_i \tilde{W}_i]$  и нижнюю границу  $E[\min_i \tilde{Y}_i]$ . В [66] рассматриваются два подхода: в первом предполагается, что по-разному распределенные величины  $\tilde{W}_i$  и  $\tilde{Y}_i$  являются полностью зависимыми, а во втором — полностью независимыми по-разному распределенными (НРР). Мы используем второй подход, поскольку он приводит к построению более точных границ. Верхняя граница  $E[\min_i \tilde{W}_i]$  задана выражением

$$E[\min_i \tilde{W}_i] = \int_0^\infty \left( \prod_{i \in \mathbb{F}_{2^m}} P\{\tilde{W}_i \geq -w\} + \prod_{i \in \mathbb{F}_{2^m}} P\{\tilde{W}_i > w\} - 1 \right) dw. \quad (3.27)$$

Поскольку случайные величины  $\tilde{W}_i$  являются Гауссовскими, получаем  $P\{\tilde{W}_i \leq w\} = 1 - \frac{1}{2} \operatorname{erfc}\left(\frac{w - \mu_i}{\sqrt{2\sigma_{\tilde{W}_i}^2}}\right)$ , где  $\sigma_{\tilde{W}_i}^2$  — дисперсия  $\tilde{W}_i$ . Аналогичное выражение может быть использовано для вычисления  $E[\min_i \tilde{Y}_i]$ , при этом  $\sigma_{\tilde{W}_i}^2$  заменяется на  $\sigma_{\tilde{Y}_i}^2$ . Необходимо найти такие дисперсии  $\sigma_{\tilde{W}_i}^2$  и  $\sigma_{\tilde{Y}_i}^2$ , при которых границы (3.26), вычисленные согласно (3.27), оказываются достаточно точными.

Определим центрированные случайные величины  $\overline{L}_i^{(b)} = l_i^{(b,\gamma)} - \rho_i^{(b,\gamma)}$ ,  $i \in \mathbb{F}_{2^m}$ ,  $b \in \{0, 1\}$ . Заметим, что дисперсии и ковариации этих случайных величин могут быть вычислены согласно выражению (3.22). Пусть  $\Sigma^{(b,\gamma)}$  — ковариационная матрица для  $l^{(b,\gamma)}$ ,  $b \in \{0, 1\}$ . Дисперсия случайной величины  $(\overline{X}_i - \overline{X}_j)$  может

быть представлена как

$$\begin{aligned}
b_{i,j} &= D[\bar{X}_i - \bar{X}_j] = D[\bar{L}_i^{(0)} + \bar{L}_{i\oplus r}^{(1)} - \bar{L}_j^{(0)} - \bar{L}_{j\oplus r}^{(1)}] = \\
&= \Sigma_{i,i}^{(0,\gamma)} + \Sigma_{i\oplus r,i\oplus r}^{(1,\gamma)} + \Sigma_{j,j}^{(0,\gamma)} + \Sigma_{j\oplus r,j\oplus r}^{(1,\gamma)} - 2(\Sigma_{i,j}^{(0,\gamma)} + \Sigma_{i\oplus r,j\oplus r}^{(1,\gamma)}) = \\
&= 2\rho_i^{(0,\gamma)} + 2\rho_{i\oplus r}^{(1,\gamma)} + 2\rho_j^{(0,\gamma)} + 2\rho_{j\oplus r}^{(1,\gamma)} - \\
&\quad - 2(\rho_i^{(0,\gamma)} + \rho_j^{(0,\gamma)} - \rho_{i\oplus j}^{(0,\gamma)} + \rho_{i\oplus r}^{(1,\gamma)} + \rho_{j\oplus r}^{(1,\gamma)} - \rho_{i\oplus j}^{(1,\gamma)}) = \\
&= 2\rho_{i\oplus j}^{(0,\gamma)} + 2\rho_{i\oplus j}^{(1,\gamma)}.
\end{aligned}$$

Условие (3.25) можно выразить через  $b_{i,j}$  как  $\sigma_{\tilde{W}_i}^2 + \sigma_{\tilde{W}_j}^2 \leq b_{i,j}$  и  $\sigma_{\tilde{Y}_i}^2 + \sigma_{\tilde{Y}_j}^2 \geq b_{i,j}$ , поскольку в случае НРР величин  $\tilde{W}_i$  и  $\tilde{Y}_i$  их ковариации равны нулю.

Для ЗРР величин  $\tilde{X}_i$  наименьшая верхняя граница для  $E[\min_i \tilde{X}_i]$  с использованием НРР величин  $\tilde{W}_i$  может быть получена как

$$B^{(Upper)} = \min_{\sigma_{\tilde{W}_i}^2 + \sigma_{\tilde{W}_j}^2 \leq b_{i,j}} E[\min_i \tilde{W}_i].$$

Аналогичная наибольшая нижняя граница для  $E[\min_i \tilde{X}_i]$  равна

$$B^{(Low)} = \max_{\sigma_{\tilde{Y}_i}^2 + \sigma_{\tilde{Y}_j}^2 \geq b_{i,j}} E[\min_i \tilde{Y}_i].$$

Поскольку сложность поиска оптимальных значений  $\sigma_{W_i}$  и  $\sigma_{Y_i}$  оказывается высокой, мы предлагаем использовать значения  $\sigma_{\tilde{W}_i}^2 = \frac{1}{2} \min_j b_{i,j}$ ,  $\sigma_{\tilde{Y}_i}^2 = \frac{1}{2} \max_j b_{i,j}$ , которые гарантированно удовлетворяют необходимым условиям. Такие субоптимальные значения  $\sigma_{W_i}$  обеспечивают достаточно хорошую точность для верхней границы. Однако значение нижней оказывается чрезмерно малым в случае существенно отличающихся  $b_{i,j}$ , а точнее, в случае наличия  $b_{i,j}$ , значения которых существенно превосходят  $\min_j b_{i,j}$ . Заметим, что исключение из рассмотрения  $l_{i\oplus j}^{(0,\gamma)}$  или  $l_{i\oplus j}^{(1,\gamma)}$  с наибольшим значениями математических ожиданий ведет к снижению  $\sigma_{\tilde{Y}_i}^2$ . При этом, исключение  $l_{i\oplus j}^{(0,\gamma)}$  или  $l_{i\oplus j}^{(1,\gamma)}$  ведет также к исключению  $\tilde{X}_{i\oplus j}$  или  $\tilde{X}_{i\oplus j\oplus r}$ , что оказывает лишь незначительное влияние на значение  $E[\min_h \tilde{X}_h]$ , поскольку математические ожидания  $\mu_{i\oplus j}$  и  $\mu_{i\oplus j\oplus r}$  существенно превосходят  $\min_h \mu_h$ . Исходя из вышесказанного, предлагается выбрать

соответствующий коэффициент  $\alpha$  и выполнить поиск минимума (3.26) только по  $i \in \Gamma$ ,  $\Gamma = \{t | \mu_t \leq \alpha \mu_{min}\}$ ,  $\mu_{min} = \min_i \mu_i$ . То есть, мы используем аппроксимацию

$$E[\min_{i \in \mathbb{F}_{2^m}} \tilde{X}_i] \approx E[\min_{i \in \Gamma} \tilde{X}_i] \geq E[\max_{i \in \Gamma} \tilde{Y}_i].$$

К примеру, при моделировании использовалось  $\alpha = 3$ .

Существуют различные пути получения оценки для  $\xi_r^{(2\gamma)} = E[\min_i \tilde{X}_i]$  с использованием верхней и нижней границ. Оказывается, что самое простое решение, среднее арифметическое, обеспечивает приемлемую точность оценкам. Таким образом, мы предлагаем заменить (3.23) на  $\rho_r^{(2\gamma)} \approx \xi_r^{(2\gamma)} - \xi_0^{(2\gamma)}$ , где

$$\xi_r^{(2\gamma)} = \frac{1}{2}(E[\min_{i \in \mathbb{F}_{2^m}} \tilde{W}_i] + E[\min_{i \in \Gamma} \tilde{Y}_i])$$

#### 3.5.4. Декодирование двоичного образа кода

В случае передачи двоичного образа кода Рида-Соломона по каналу без памяти можно показать, что

$$P(u_0^i | y_0^{n-1}) = \prod_{j=0}^{m-1} P(u_0^i[j] | y_0^{n-1}[j]), \quad (3.28)$$

$$R_{2^m}(u_0^i, y_0^{n-1}) = \prod_{j=0}^{m-1} R(u_0^i[j], y_0^{n-1}[j]), \quad (3.29)$$

где  $u_0^i[j] = (u_0[j], \dots, u_i[j])$ ,  $u_s = \sum_{j=0}^{m-1} u_s[j] a_j$ ,  $u_s[j] \in \mathbb{F}_2$ ,  $(a_0, \dots, a_{m-1})$  — некоторый базис поля  $\mathbb{F}_{2^m}$ , и  $y_0^{n-1}[j]$  — последовательность, соответствующая  $j$ -ым битам символов принятой последовательности  $y_0^{n-1}$ . Таким образом, декодирование кода над  $\mathbb{F}_{2^m}$  может быть реализовано посредством  $m$  синхронизированных запусков декодера для двоичных кодов. Синхронизация необходима для вычисления  $R_{2^m}(u_0^i, y_0^{n-1})$  по найденным  $R(u_0^i[j], y_0^{n-1}[j])$ ,  $0 \leq j < m$ , на слое поляризирующего преобразования, соответствующем входной последовательности, а также вычисления значений динамически замороженных символов согласно выражению (2.1) с использованием арифметики поля  $\mathbb{F}_{2^m}$ . Применение этого подхода



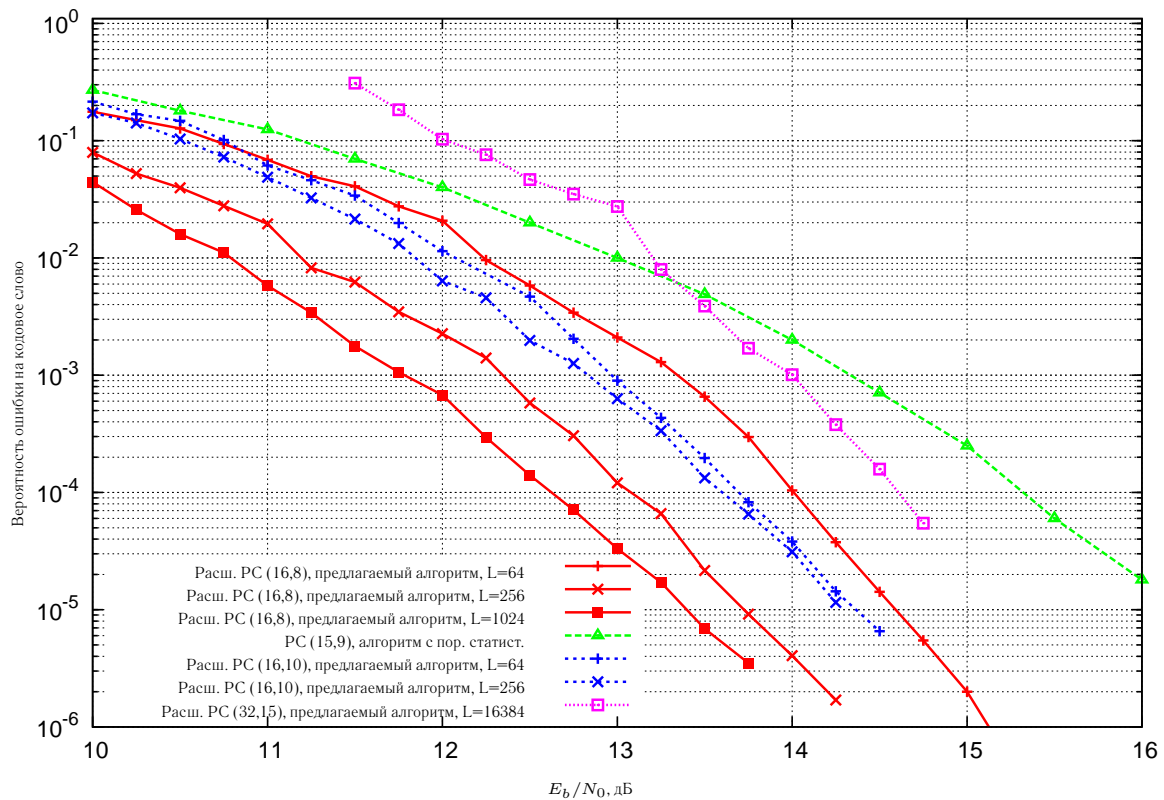


Рис. 3.14. Вероятность ошибки декодирования кодов Рида-Соломона над  $\mathbb{F}_{2^m}$  в аддитивном Гауссовском канале при  $2^m$ -КАМ

в случае передачи двоичного образа приводит к значительному снижению сложности декодирования.

Выражение (3.28) предполагает, что вероятность ошибки в  $i$ -ом символьном подканале может быть вычислена как  $P_i^{(2^m)} = 1 - (1 - P_i)^m$ , где  $P_i$  — вероятность ошибки декодирования бита, вычисленная как вероятность ошибки в  $i$ -ом битовом подканале поляризирующего преобразования с двоичным входным алфавитом (см. раздел 1.3.1.2). Таким образом, значения функции  $\hat{\Omega}(i)$  могут быть вычислены с помощью методов, разработанных для двоичных полярных кодов [73, 76].

### 3.5.5. Численные результаты

На Рис. 3.14 представлены результаты статистического моделирования, иллюстрирующие корректирующую способность предлагаемого алгоритма декоди-

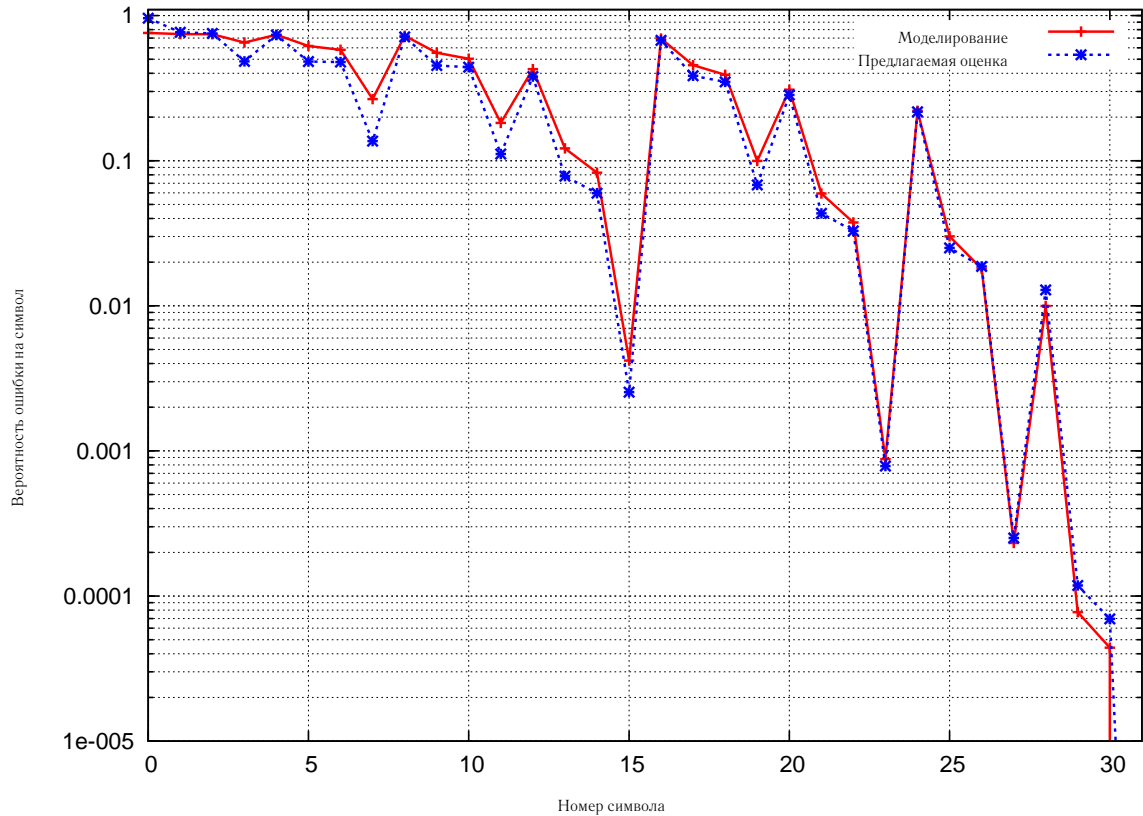


Рис. 3.15. Вероятность ошибки декодирования на символ в аддитивном Гауссовском канале при  $2^5$ -КАМ,  $N_0 = 0.08$

рования для кодов Рида-Соломона над  $\mathbb{F}_{2^m}$  и  $2^m$ -КАМ модуляции. При декодировании в кодovém дереве рассматривалось не более  $L$  путей  $u_0^{i-1}$  каждой длины  $0 \leq i < 2^m$ . Это соответствует списочному алгоритму последовательного исключения [72] с размером списка  $L$ . Заметим, что для сохранения достаточной корректирующей способности необходимо экспоненциально увеличивать  $L$  при росте  $n = 2^m$ . Видно, что предлагаемый подход обеспечивает энергетический выигрыш в 2 дБ по сравнению с алгоритмом декодирования, использующим порядковые статистики [20].

На Рис. 3.15 представлено сравнение вероятностей ошибки в подканалах  $32 \times 32$  поляризирующего преобразования, вычисленных с помощью метода, предложенного в разделе 3.5.3.2, и вероятностей, полученных в результате моделирования. Видно, что за исключением нескольких подканалов с малыми индексами, полученные оценки хорошо согласуются с экспериментальными данными.

На Рис. 3.16 приведены графики зависимости вероятности ошибки деко-

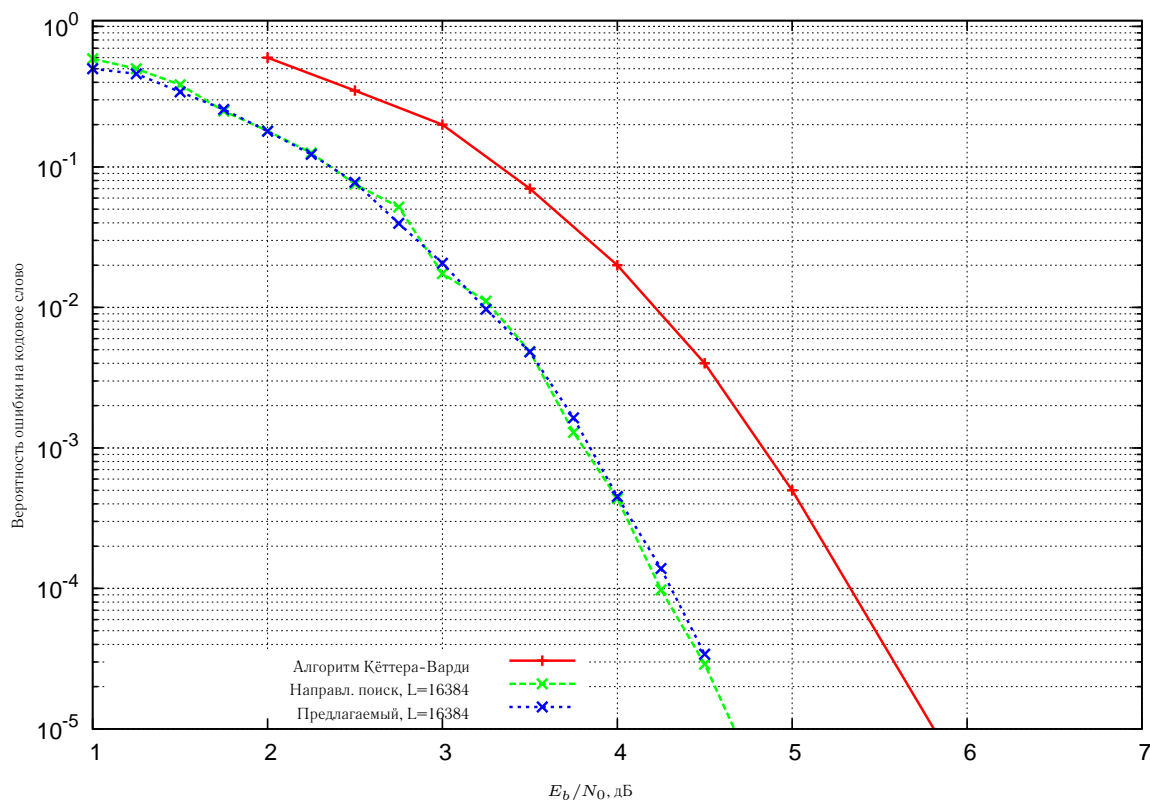


Рис. 3.16. Вероятность ошибки декодирования кода Рида-Соломона (31, 15) в случае передачи двоичного образа по аддитивному Гауссовскому каналу

дирования от отношения сигнал/шум для предлагаемого алгоритма декодирования, метода мягкого декодирования Кёттера-Варди, описанного в разделе 1.5.1, и декодирования методом направленного поиска для кодов Рида-Соломона, представленном в [9]. Рассматривается случай передачи двоичного образа кодовых слов кода Рида-Соломона (31, 15, 17) по аддитивному Гауссовскому каналу. Видно, что предлагаемый подход обеспечивает такую же корректирующую способность, как декодирование методом направленного поиска, и оба эти алгоритма обеспечивают выигрыш 1 дБ по сравнению с алгоритмом Кёттера-Варди. Рис. 3.17 иллюстрирует среднее число итераций, выполняемых предлагаемым алгоритмом и методом направленного поиска. При отношении сигнал/шум 4 дБ предлагаемый алгоритм требует в среднем 2.4 раза меньше итераций. Недостатком предлагаемого подхода является необходимость использования списка большого размера  $L$ . При этом величина  $L$ , необходимая для обеспечения приемлемой вероятности ошибки декодирования, растет экспоненциально с увели-

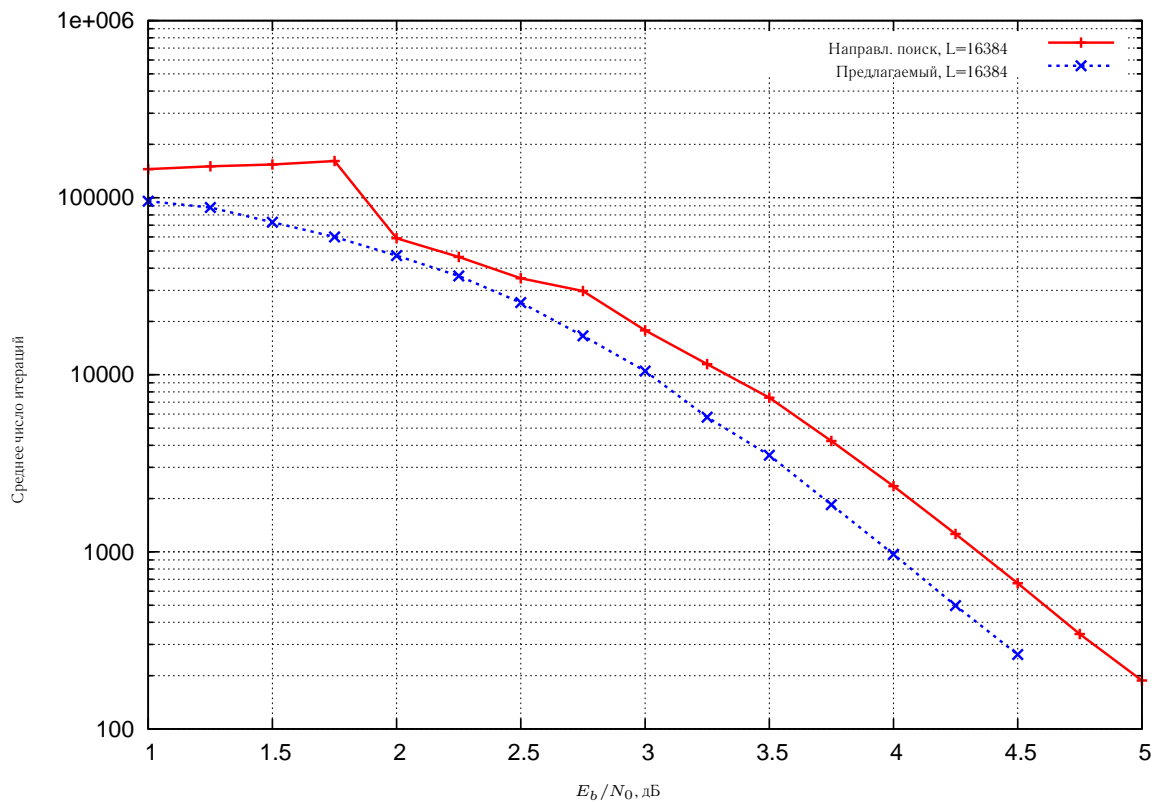


Рис. 3.17. Среднее число итераций для кода Рида-Соломона (31, 15) в случае передачи двоичного образа по аддитивному Гауссовскому каналу

чением длины кода, что ограничивает область применимости предлагаемого метода сравнительно короткими кодами.

### 3.5.6. Выводы

В данном разделе предложен новый алгоритм декодирования коротких кодов Рида-Соломона над полем  $\mathbb{F}_{2^m}$ . Предлагаемый алгоритм может рассматриваться как обобщение алгоритма, предложенного в разделе 3.2 для декодирования двоичных полярных кодов с ядром Арикана, на случай кодов над  $\mathbb{F}_{2^m}$ . Показано, что при достаточно большом размере списка предлагаемый подход обеспечивает меньшую вероятность ошибки декодирования, чем метод Кёттера-Варди и декодирование с использованием порядковых статистик. Кроме того, сложность предлагаемого метода меньше сложности декодирования методом направленного поиска.

### 3.6. Выводы по разделу

В данном разделе был представлен новый алгоритм последовательного декодирования полярных кодов с ядром Арикана. Результаты моделирования показывают, что этот алгоритм обладает существенно меньшей сложностью по сравнению с существующими списочными и стековыми алгоритмами декодирования полярных кодов. Снижение сложности достигается за счет незначительного ухудшения корректирующей способности.

Было выполнено обобщение предложенного метода последовательного декодирования на случай полярных кодов с произвольным двоичным ядром. Алгоритм предполагает использование методов декодирования по максимуму правдоподобия для кодов, порожденных строками ядра. Численные результаты показывают, что предлагаемый подход позволяет выполнять декодирование полярных кодов с ядром БЧХ почти по максимуму правдоподобия. Показано, что такие коды превосходят по корректирующей способности полярные коды с ядром Арикана с контрольной суммой CRC по крайней мере при малых отношениях сигнал/шум.

На базе предложенного метода последовательного декодирования был построен алгоритм декодирования коротких кодов Рида-Соломона над полем  $\mathbb{F}_{2^m}$ . Показано, что при достаточно большом размере списка предлагаемый подход обеспечивает меньшую вероятность ошибки декодирования, чем метод Кёттера-Варди.

## Глава 4

# Декодирование длинных кодов Рида-Соломона

Несмотря на то, что последовательный алгоритм декодирования кодов Рида-Соломона, предложенный в разделе 3.5, обладает большей корректирующей способностью по сравнению с методом Кёттера-Варди (см. раздел 1.5.1), задача разработки быстрой реализации метода Кёттера-Варди для длинных кодов Рида-Соломона остается актуальной, поскольку при использовании последовательного алгоритма объем потребляемой памяти растет экспоненциально с ростом длины кода.

Наиболее трудоемким шагом метода Кёттера-Варди является построение полинома от двух переменных такого, что точки  $(x_i, y_j)$  являются его корнями заданной кратности  $M_{i,j}$ ,  $0 \leq i < n$ ,  $0 \leq j < q$ . При этом интерполяционный полином должен иметь минимально возможную  $(1, k - 1)$ -взвешенную степень. Для реализации данного шага различными исследователями был разработан ряд алгоритмов, представленных в разделе 1.5.2. Тем не менее, сложность данных алгоритмов остается чрезмерно высокой. В данном разделе будут представлены новые быстрые алгоритмы решения задачи двумерной интерполяции.

### 4.1. Послойный алгоритм

В основе алгоритмов, описанных в разделе 1.5.2, лежит то, что множество полиномов, имеющих корни  $(x_i, y_j)$  кратности  $M_{i,j}$ , является идеалом, а также то, что искомым интерполяционным полином может быть найден в базисе Грёбнера этого идеала, построенном для  $(1, k - 1)$ -взвешенного лексикографического упорядочения.

Операция умножения, определенная на множестве идеалов полиномов с общими корнями заданной кратности, является ассоциативной, то есть  $I_1(I_2I_3) =$

$(I_1 I_2) I_3$ . Нейтральным элементом по отношению к операции умножения является идеал, совпадающий со всем кольцом многочленов. Следовательно, множество таких идеалов образует моноид. Это дает возможность разбить задачу построения идеала интерполяционных многочленов на задачи построения нескольких просто описываемых идеалов и задачу их перемножения. Данный подход может быть реализован двумя способами: путем наращивания кратностей корней, что соответствует предлагаемому алгоритму, и путем последовательного добавления корней заданной кратности, что соответствует итеративному интерполяционному алгоритму.

Для решения задачи вычисления произведения идеалов в данной работе используется рандомизированный алгоритм умножения идеалов, описанный в разделе 1.5.2.2. На основе анализа его сходимости далее будут выбраны такие алгоритмы решения подзадач задачи построения идеала интерполяционных многочленов, которые позволяют минимизировать общую вычислительную сложность.

В основе предлагаемого послойного алгоритма двумерной интерполяции лежит двоичное разложение элементов матрицы кратностей корней. Матрица кратностей корней может быть представлена в следующем виде

$$M = \sum_{h=0}^m 2^h M^{(h)}, \quad (4.1)$$

где  $m = \max_{i,j} [\log M_{i,j}]$ , и  $M^{(h)}$  — матрица локаторов корней из  $h$ -го слоя<sup>1</sup>, состоящая из  $\{1, 0\}$ .  $h$ -м слоем  $L_h$  назовем множество точек  $(x_i, y_j)$  для которых  $M_{i,j}^{(h)} = 1$ .

Матрица  $M^{(h)}$  может быть представлена как  $M^{(h)} = \sum_{t=1}^{g_h} M^{(h,t)}$ , где каждая строка матрицы  $M^{(h,t)}$  содержит не более одного ненулевого элемента, и  $g_h$  — максимальное количество единиц в строке матрицы  $M^{(h)}$ . Таким образом, слой  $L_h$  разбивается на  $g_h$  групп, причем каждая группа  $L_{h,t}$  содержит точки  $(x_i, y_j)$  с

---

<sup>1</sup> Заметим, что эти слои не связаны со слоями поляризующего преобразования.

различными  $x_i$ .

Для каждой группы  $L_{h,t}$  можно построить идеал

$$I_{M^{(h,t)}} = \{Q(x, y) \mid Q(x_i, y_j) = 0, (x_i, y_j) \in L_{h,t}\}.$$

Идеал полиномов, имеющих корни в точках слоя  $L_h$

$$I_{M^{(h)}} = \{Q(x, y) \mid Q(x_i, y_j) = 0, (x_i, y_j) \in L_h\},$$

может быть получен, как  $I_{M^{(h)}} = \prod_{t=1}^{g_h} I_{M^{(h,t)}}$ . При обобщении критерия останова (1.22) рандомизированного алгоритма умножения идеалов *Merge* на случай корней различных кратностей  $M'_{i,j}$ , получаем

$$\Delta(\mathcal{B}) = \Delta_{M'}, \quad (4.2)$$

где

$$\Delta_{M'} = \sum_{i=0}^{n-1} \sum_{j=0}^{|\mathbb{F}|-1} \frac{M'_{i,j}(M'_{i,j} + 1)}{2}.$$

Идеал полиномов, кратности корней которых соответствуют элементам матрицы  $M$ , вычисляется согласно формуле

$$I_M = (\dots (I_{M^{(m)}}^2 \cdot I_{M^{(m-1)}})^2 \cdot I_{M^{(m-2)}} \dots I_{M^{(1)}})^2 \cdot I_{M^{(0)}},$$

где  $I^2 = I \cdot I$ ,  $I^0 = \mathbb{F}_q[x, y]$  и  $I \cdot \mathbb{F}_q[x, y] = I$ .

Также данный подход может быть реализован следующим образом

$$J_{h+1} = K_h I_{M^{(m-h-1)}}, K_h = J_h^2, J_0 = I_{M^{(m)}} \quad (4.3)$$

где  $I^2 = I \cdot I$ . Пусть  $\rho_h$  — наименьшее целое число такое, что базис Грёбнера модуля  $J_{h,\rho_h}$  также является базисом для  $J_h$ . Заметим, что  $J_{h,\rho_h} = I_{W^{(h),\rho_h}}$ , где  $W^{(h)} = \sum_{t=0}^h 2^t M^{(m-t)}$ .

#### 4.1.1. Построение базиса Грёбнера идеала группы

Алгоритм для вычисления базиса Грёбнера идеала  $I_{M^{(h,t)}}$  представлен на Рис. 4.1. Данный алгоритм предполагает, что каждая строка матрицы  $M^{(h,t)}$  содержит не более одной единицы.



В статье [75] было показано, что идеал полиномов, имеющих корни в точках  $(x_i, y_i)$  с различными значениями  $x_i$ , порождается  $I = \langle \phi(x), y - \Upsilon(x) \rangle$ , где  $\phi(x) = \prod_i (x - x_i)$  и  $\Upsilon(x_i) = y_i$ . Такие полиномы строятся на первых двух шагах данного алгоритма, где  $x_i$  и  $y_i$  соответствует ненулевым элементам матрицы локаторов корней  $M$ . В ходе работы алгоритма *GroupBasis* происходит построение последовательности модулей

$$\mathcal{M}^{(h+1)} = \{Q(x, y) + a(x)y^h(y - \Upsilon(x)) \mid Q(x, y) \in \mathcal{M}^{(h)}\}, h \geq 0,$$

где  $\mathcal{M}^{(0)} = [\phi(x)]$ . Для того, чтобы для каждого  $h$  получить базис Грёбнера модуля по отношению к  $(1, k - 1)$ -взвешенной степени лексикографического упорядочения, используется алгоритм *Reduce*. Как только получается модуль, базис Грёбнера которого содержит полином  $B_h(x, y) : \text{LT } B_h(x, y) = y^{h-1}$ , выполнение шагов цикла *REPEAT* (строки 5–8) может быть остановлено, поскольку полученный базис Грёбнера является одним из базисов идеала  $I = \langle \phi(x), y - \Upsilon(x) \rangle$ . Данное утверждение основано на том, что базис Грёбнера нульмерного идеала должен содержать полином  $B(x, y) : \text{LT } B(x, y) = y^h$  для некоторого  $h$ , что было показано в [15].

#### 4.1.2. Переход от идеалов групп к идеалу слоя

Идеал полиномов, имеющих корни во всех точках  $h$ -го слоя, может быть получен путем перемножения идеалов, построенных для соответствующих групп.

Алгоритм, псевдокод которого представлен на Рис. 4.2, производит разбиение слоя на  $g$  групп в соответствии с матрицей локаторов корней  $M^{(h)}$ . Напомним, что каждая группа состоит из точек  $(x_i, y_j)$  с различными  $x_i$ , такими, что  $M_{i,j}^{(h)} = 1$ . В первые группы войдут точки  $(x_i, y_j)$  с наиболее часто встречающимися  $x_i$ . Для каждой группы алгоритм *GroupBasis* возвращает базис Грёбнера идеала. Для перемножения идеалов, соответствующих полученным группам, используется алгоритм *Merge*.

GROUPBASIS( $M$ )

```

1   $\phi(x) \leftarrow \prod_{i:M_{i,j}=1} (x - x_i)$ 
2   $\Upsilon(x) \leftarrow \sum_{i,j:M_{i,j}=1} y_j \frac{\prod_{i':M_{i',j}=1, i' \neq i} (x - x_{i'})}{\prod_{i':M_{i',j}=1, i' \neq i} (x_i - x_{i'})}$ 
3   $\mathcal{B} \leftarrow (\phi(x))$ 
4   $h \leftarrow 0$ 
5  repeat
6       $\mathcal{B} \leftarrow \text{REDUCE}(\mathcal{B}, y^h(y - \Upsilon(x)))$ 
7       $h \leftarrow h + 1$ 
8  until  $\text{LT } \mathcal{B}_h = y^{h-1}$ 
9  return  $\mathcal{B}$ 

```

Рис. 4.1. Построение базиса Грёбнера для  $I_{M^{(h,t)}}$

LAYERBASIS( $M$ )

```

1   $\mathcal{B} \leftarrow (1)$ 
2  for  $i \leftarrow 0$  to  $n - 1$ 
3  do  $c_i = \text{wt}(M_{i,-})$ 
4   $g \leftarrow \max_i(c_i)$ 
5  for  $t \leftarrow g$  to 1
6  do  $M^{(t)} = \mathbf{0}$ 
7  for  $i \leftarrow 1$  to  $n$ 
8  do if  $c_i \geq t$ 
9  then Выбрать  $j : M_{i,j} = 1$ 
10      $M_{i,j}^{(t)} = 1$ 
11      $M_{i,j} = 0$ 
12   $\mathcal{G} \leftarrow \text{GROUPBASIS}(M^{(t)})$ 
13   $\mathcal{B} \leftarrow \text{MERGE}(\mathcal{B}, \mathcal{G})$ 
14 return  $\mathcal{B}$ 

```

Рис. 4.2. Построение базиса Грёбнера для  $I_{M^{(h)}}$

```

LAYEREDINTERPOLATION( $M$ )
1   $m \leftarrow \max_{i,j}(\log M_{i,j})$ 
2  Let  $M = \sum_{h=0}^m 2^h M^{(h)}$ ,  $M_{i,j}^{(h)} \in \{0, 1\}$ 
3   $\mathcal{B} \leftarrow \text{LAYERBASIS}(M^{(m)})$ 
4  for  $h \leftarrow m - 1$  to 0
5  do  $\mathcal{B} \leftarrow \text{MERGE}(\mathcal{B}, \mathcal{B})$ 
6     $\mathcal{G} \leftarrow \text{LAYERBASIS}(M^{(h)})$ 
7     $\mathcal{B} \leftarrow \text{MERGE}(\mathcal{B}, \mathcal{G})$ 
8  return  $\mathcal{B}$ 

```

Рис. 4.3. Построение базиса Грёбнера для  $I_M$

### 4.1.3. Интерполяция в случае корней переменной кратности

Послойный интерполяционный алгоритм для случая корней переменной кратности представлен на Рис. 4.3. Алгоритм использует побитовую декомпозицию матрицы кратностей корней  $M$ , заданную равенством (4.1), для построения последовательности базисов Грёбнера идеалов  $I_{M^{(h)}}$ . А именно, для каждого значения  $h$  он вызывает *Merge* для вычисления  $I_{M^{(h+1)}}^2$  (строка 5), строит базис Грёбнера идеала полиномов с корнями во всех точках  $h$ -го слоя (строка 6), и находит  $I_{M^{(h)}}$  после вызова *Merge* (строка 7).

## 4.2. Сходимость рандомизированного алгоритма умножения идеалов

Покажем, что сложность умножения идеалов с помощью рандомизированного алгоритма, заданного формулой (1.24), существенно зависит от свойств умножаемых идеалов. Сложность построения базиса Грёбнера  $I_{M_1}I_{M_2}$  зависит от длины последовательности подмодулей  $\mathcal{N}^{(0)} \supset \mathcal{N}^{(1)} \supset \dots$ , имеющих базисы Грёбнера  $\mathcal{B}^{(0)}, \mathcal{B}^{(1)}, \dots$ , такие что  $\Delta(\mathcal{B}^{(i)}) > \Delta_{M_1+M_2}$ . Длина данной последовательности пропорциональна  $N = \Delta(\mathcal{B}^{(i)}) - \Delta_{M_1+M_2}$ . Сложность многомерного

алгоритма Евклида также возрастает при увеличении  $N$ .

Не теряя общности, рассмотрим случай корней фиксированной кратности (случай списочного декодирования). В [75] было показано, что в случае  $M_i = r_i H$ ,  $i = 1, 2$ , где  $r_i \in \mathbb{N}$ ,  $H - n \times |\mathbb{F}|$  матрица, содержащая ровно одну единицу в каждой строке,

$$N \approx \frac{n}{2} \left( \frac{\rho_2}{\rho_1 + 1} r_1 (r_1 + 1) + \frac{\rho_1}{\rho_2 + 1} r_2 (r_2 + 1) - 2r_1 r_2 \right)$$

где  $\rho_i$  - наибольшая степень по  $y$  полиномов базисов умножаемых идеалов. В статье [61] было показано, что

$$\rho_i(\rho_i + 1) \leq nr_i(r_i + 1)/(k - 1) < (\rho_i + 1)(\rho_i + 2),$$

то есть  $\rho_i + 1 \approx \alpha r_i$  для некоторого  $\alpha$ . Отсюда следует, что

$$N \approx \frac{n}{2\alpha} ((\alpha - 1)(r_1 + r_2) - 2).$$

В случае  $r_1 = r_2 = r/2$

$$N \approx \frac{\nu}{\alpha} ((\alpha - 1)r - 1). \quad (4.4)$$

Аналогичное значение можно получить в случае  $r_1 = r, r_2 = 1$ . Таким образом, сложность умножения идеалов приблизительно одинаковая для  $r_1 = r_2 = r/2$  и для  $r_1 = r, r_2 = 1$ . В случае корней переменной кратности сложность вычисления базисов Грёбнера  $K_h = J_h^2$  и  $J_{h+1} = K_h I_{M^{(m-h-1)}}$  также примерно одинаковая. Поскольку большинство матриц  $M^{(h)}$  являются разреженными, то количество операций, необходимых для построения  $J_{h+1}$  из  $K_h$  посредством умножения идеалов, является непропорционально большим.

Рассмотрим случай взаимно простых идеалов  $I_{M_1}$  and  $I_{M_2}$ , где  $M_i = rH_i$ , и  $H = H_1 + H_2$  - матрица, содержащая только одну единицу в каждой строке. Для простоты предположим, что  $\rho_1 = \rho_2 = \rho$  и что матрицы  $M_1$  и  $M_2$  имеют  $\nu$  ненулевых элементов. Тогда  $\Delta_{M_1} = \Delta_{M_2} = \nu r(r + 1)/2$  и  $\Delta_{M_1+M_2} = \nu r(r + 1)$ . Пусть  $(V_0(x, y), \dots, V_\rho(x, y))$  и  $(S_0(x, y), \dots, S_\rho(x, y))$  являются такими базисами Грёбнера для  $I_{M_1}$  и  $I_{M_2}$ , что  $\text{LT } V_i(x, y) = \beta_i x^{\omega_i} y^i$  и  $\text{LT } S_i(x, y) = \gamma_i x^{s_i} y^i$ . В статье

[75] было показано, что  $\omega_i \approx s_i \approx l_r - j(k - 1)$ , где  $l_r \approx ((k - 1)\rho(\rho + 1) + \nu r(r + 1))/(2(\rho + 1))$ . Предполагая, что базис Грёбнера  $\mathcal{B}^{(0)}$  модуля  $\mathcal{N}^{(0)}$  задается формулой (1.23), можно получить оценку  $\Delta(\mathcal{B}^{(0)}) = \sum_{j=0}^{2\rho} \text{xdeg } Q_j(x, y) \approx \frac{\nu}{\alpha}(r + 1)(2\alpha r - 1)$ . Отсюда следует, что

$$N \approx \frac{\nu}{\alpha}(r + 1)(\alpha r - 1). \quad (4.5)$$

Из сравнения (4.5) и (4.4) следует, что сходимость рандомизированного алгоритма умножения идеалов намного медленнее в случае взаимно простых идеалов. Таким образом, следует избегать умножений взаимно простых идеалов, а также идеалов  $I_{M_1}$  и  $I_{M_2}$  с существенно отличающимися  $M_1$  и  $M_2$ . Точнее, следует избегать вычислений  $J_{h+1} = K_h I_{M^{(m-h-1)}}$ . Кроме того, метод построения  $I_{M^{(h)}}$ , представленный на Рис. 4.2, должен быть заменен на более эффективный.

Решения данных проблем будут рассмотрены в следующем разделе.

### 4.3. Построение базиса для $M^{(m)}$

В большинстве случаев матрица кратностей корней  $M^{(m)}$  содержит много ненулевых элементов. Мы предлагаем использовать модифицированный алгоритм Ли-О’Салливана для уменьшения сложности построения базиса Грёбнера модуля, соответствующего матрице  $M^{(m)}$ , который является начальной точкой для послойного алгоритма. Для простоты изложения индекс слоя  $m$  будет опущен. Напомним, что для снижения сложности интерполяции используется метод перекодирования, описанный в разделе 1.5.3. Предположим, что группа  $G_0$  содержит все локаторы перекодирования, принадлежащие слою  $L$ . После применения преобразования перекодирования полиномы базиса (1.25) могут быть представлены как

$$\widehat{B}_0(x, z) = \prod_{i \in \{L \setminus R\}_x} (x - x_i), \quad (4.6)$$

$$\begin{aligned}
\widehat{B}_s(x, z) &= u_s(x) \frac{\phi(x)z + f(x) - w_0(x)}{\psi^{(h)}(x)} a_s(x, z) = \\
&= \left( \sum_{(x_i, y_j) \in R \setminus G_0} \frac{y_j \lambda_{R \setminus G_0}(x)}{\phi'(x_i)(x - x_i)} - \sum_{(x_i, y_j) \in G_0 \setminus R} \frac{y_j \lambda_{G_0 \setminus R}(x)}{u_0'(x_i)(x - x_i)} + \right. \\
&\quad \left. + \sum_{(x_i, y_j) \in R \cap G_0} y_j \frac{\frac{\lambda_{R \setminus G_0}(x)}{\phi'(x_i)} - \frac{\lambda_{G_0 \setminus R}(x)}{u_0'(x_i)}}{x - x_i} \right) u_s(x) a_s(x, z), \quad s \geq 1.
\end{aligned}$$

где  $a_s(x, z) = \prod_{t=1}^{s-1} (\phi(x)z + f(x) - w_t(x))$  и  $\lambda_A(x) = \prod_{(x_i, y_j) \in A} (x - x_i)$ . Заметим, что данные выражения являются более простыми, чем выражения, приведенные в [51].

Данные полиномы следует обработать согласно процедуре, описанной в разделе 1.5.2.3, используя упорядочение  $\prec_{-1}$ . Критерий останова на 4-ом шаге должен быть заменен на

$$\text{xdeg } \widehat{B}_s(x, z) = sk - \tau_m. \quad (4.7)$$

Данный метод требует  $\rho \approx \theta$  вызовов многомерного алгоритма Евклида, в то время как каждой операции умножения взаимно простых идеалов, Рис. 4.2, обычно требуется больше одного вызова, а количество таких умножений равно  $\theta$ .

#### 4.4. Увеличение кратностей корней

Анализ, приведенный в разделе 4.2, предполагает, что вычисление базисов Грёбнера для  $J_{h+1} = K_h I_{M^{(m-h-1)}}$  и  $K_h = J_h^2$  посредством умножения идеалов требуют приблизительно одинаковое число операций. Вычисление  $J_{h+1} = K_h I_{M^{(m-h-1)}}$  является неэффективным, поскольку обычно матрицы  $M^{(m-h-1)}$  разреженные. Мы предлагаем реализовать данный шаг с помощью итеративного интерполяционного алгоритма [40]. Однако перед применением данного алгоритма необходимо обработать точки  $(x_i, y_j) \in R \cap L_h$ . Без перекодирования мы могли увеличить кратности данных корней путем умножения  $K_{h, \rho'_h}$  (для некоторого  $\rho'_h$ )

на  $\Upsilon_h = [\psi^{(h)}(x), y - w_{h,0}(x)]$ , где  $w_{h,0}(x_i) = y_j$  для всех  $(x_i, y_j) \in R \cap L_h$ . Заметим, что данный базис является базисом Грёбнера по отношению к  $\prec_{k-1}$ . После замены переменных он преобразуется в  $[\psi^{(h)}(x), \phi(x)z + f(x) - w_{h,0}(x)] = [\psi^{(h)}(x), \phi(x)z]$ . Исключив общий множитель  $\psi^{(h)}(x)$ , мы получим модуль  $\widehat{\Upsilon}_h = [1, \prod_{(x_i, y_j) \in R, M_{i,j}^{(h)}=0} (x - x_i)z]$ . Тогда  $\widehat{\mathcal{J}}'_{h+1} = \widehat{K}_{h, \rho'_h} \widehat{\Upsilon}_h$ , где  $\widehat{K}_{h, \rho'_h}$  является модулем, полученным из  $K_{h, \rho'_h}$  в результате применения преобразования перекодирования. Учитывая, что базис  $\widehat{\Upsilon}_h$  содержит лишь два полинома, применение рандомизированного алгоритма умножения идеалов, заданного (1.24), оказывается эффективным.

Остальные точки  $(x_i, y_j) \notin R : M_{i,j}^{(m-h-1)} = 1$  могут быть обработаны с помощью итеративного интерполяционного алгоритма. То есть, для всех  $j_1, j_2 : j_1 + j_2 + 1 = 2W_{i,j}^{(h)} + M_{i,j}^{(m-h-1)}$ , следует применить шаги 5–12 алгоритма *IterativeInterpolate*, представленного на Рис. 1.14, к базису Грёбнера  $\widehat{\mathcal{J}}'_{h+1}$ . Заметим, что нам необходимо построить (до перекодирования) полиномы  $Q_t(x, y), t = 0, \dots, \rho_{h+1}$ , которые составляют базис Грёбнера как модуля  $I_{W^{(h+1)}, \rho_{h+1}}$ , так и нульмерного идеала  $I_{W^{(h+1)}}$ . Это предполагает, что  $\text{xdeg } Q_{\rho_{h+1}}(x, y) = 0$ . После перекодирования данное ограничение преобразуется в (4.7). Однако при определенных условиях итеративным интерполяционным алгоритмом на 9-ом шаге может быть выбран последний полином в качестве наименьшего, который далее умножается на  $x - \alpha$  на 12-ом шаге, что приводит к нарушению ограничения на  $\text{xdeg}$ . Для разрешения данной проблемы необходимо добавить в базис еще один полином, равный последнему полиному, умноженному на  $\phi(x)z$  в случае использования преобразования перекодирования.

```

HYBRIDINTERPOLATION( $M$ )
1   $R \leftarrow \left\{ \arg \max_{(x_i, y_j)} M_{i,j} \right\}, \quad |R| = k$ 
2   $m \leftarrow \left\lceil \log \max_{i,j} M_{i,j} \right\rceil$ 
3  Let  $M = \sum_{h=0}^m 2^h M^{(h)}, M_{i,j}^{(h)} \in \{0, 1\}$ 
4   $\mathcal{B} \leftarrow \text{INITIALMODULE}(M^{(m)}, R)$ 
5  for  $h \leftarrow m - 1$  to 0
6  do  $\mathcal{B} \leftarrow \text{MERGE}(\mathcal{B}, \mathcal{B})$ 
7   $\mathcal{G} \leftarrow (1, \prod_{(x_i, y_i) \in R, M_{i,j}^{(h)} = 0} (x - x_i)z)$ 
8   $\mathcal{B} \leftarrow \text{MERGE}(\mathcal{B}, \mathcal{G})$ 
9  for  $(x_i, y_j) \notin R$ 
10 do for  $j_1, j_2 : j_1 + j_2 + 1 = \sum_{t=h}^m 2^{t-h} M_{i,j}^{(t)}$ 
11   do  $\rho \leftarrow |\mathcal{B}| - 1$ 
12      $\sigma_t \leftarrow \text{HASSE}(B_t, x_i, y_j, j_1, j_2, R), t = 0, \dots, \rho$ 
13      $t_0 \leftarrow \arg \min_{t: \sigma_t \neq 0} \text{wdeg}_{(1, -1)} \text{LT } Q_t(x, z)$ 
14     if  $t_0 = \rho$ 
15       then  $B_{\rho+1} = B_\rho \phi(x)z$ 
16          $\sigma_{\rho+1} \leftarrow \text{HASSE}(B_{\rho+1}, x_i, y_j, j_1, j_2, R)$ 
17          $\rho = \rho + 1$ 
18          $B_t(x, z) \leftarrow B_t(x, z) - \frac{\sigma_t}{\sigma_{t_0}} B_{t_0}(x, z), t \neq t_0$ 
19          $B_{t_0}(x, z) \leftarrow B_{t_0}(x, z)(x - x_i)$ 
20 return  $\mathcal{B}$ 

```

Рис. 4.4. Построение базиса Грёбнера идеала  $I_M$



Таблица 4.1. Время интерполяции для (63, 31)-кода, с

$E_b/N_0$	$\max M_{i,j} = 3$		$\max M_{i,j} = 6$		$\max M_{i,j} = 9$		$\max M_{i,j} = 12$		$\max M_{i,j} = 15$	
	НА	ПА	НА	ПА	НА	ПА	НА	ПА	НА	ПА
4.8	0.0014	0.0050	0.0194	0.0419	0.0695	0.2009	0.2847	0.637	0.6845	1.556
5.0	0.0016	0.0056	0.0183	0.0450	0.0732	0.2166	0.2986	0.656	0.6915	1.596
5.2	0.0017	0.0059	0.0165	0.0471	0.0770	0.2141	0.3207	0.653	0.7070	1.605
5.4	0.0018	0.0064	0.0145	0.0473	0.0830	0.2167	0.3199	0.664	0.7108	1.557
5.6	0.0021	0.0066	0.0135	0.0492	0.0860	0.2237	0.3037	0.638	0.7121	1.470

## 4.5. Гибридный алгоритм

### 4.5.1. Описание алгоритма

На Рис. 4.4 представлен предлагаемый алгоритм. Функция *InitialModule* реализует метод построения идеала слоя, описанный в разделе 4.3. Алгоритм *Merge* подробно описан в [75]. Функция  $Hasse(B_t, x_i, y_j, j_1, j_2, R)$  предназначена для вычисления соответствующих производных Хассе полинома  $B_t$  при  $i \notin \{R\}_x$ , и модифицированных производных Хассе (см. [40, выражение (52)]), учитывающих выполненную замену переменных, при  $i \in \{R\}_x$ .

Сложность предложенного алгоритма задается сложностью обработки последнего слоя. Согласно [75], при возведении модулей в квадрат требуется  $O(n\mu^3(a \log(\mu\sqrt{nk}) \log(n\mu) + b(n - \sqrt{nk})))$  операций, где  $\mu$  — наибольшая кратность корней и  $a, b$  — некоторые коэффициенты. Учитывая, что  $\mu \sim \rho$ , вычислительная сложность, соответствующая 9 – 18 строкам алгоритма на Рис. 4.4, может быть оценена как  $O(n^2\mu^4)$ . Таким образом, общая сложность гибридного интерполяционного алгоритма равна  $O(n^2\mu^4)$ , что меньше сложности итеративного интерполяционного алгоритма  $O(n^2\mu^5)$ .

Таблица 4.2. Время интерполяции для (255, 239)-кода, с

$E_b/N_0$	$\max M_{i,j} = 3$		$\max M_{i,j} = 6$		$\max M_{i,j} = 9$		$\max M_{i,j} = 12$		$\max M_{i,j} = 15$	
	НА	ПА	НА	ПА	НА	ПА	НА	ПА	НА	ПА
6.0	0.0077	0.0139	0.0161	0.0273	0.0394	0.0690	0.1835	0.168	0.2160	0.388
6.2	0.0063	0.0139	0.0122	0.0264	0.0356	0.0651	0.1447	0.157	0.1934	0.357
6.4	0.0060	0.0141	0.0109	0.0264	0.0316	0.0628	0.0921	0.142	0.1648	0.315
6.6	0.0059	0.0138	0.0108	0.0259	0.0281	0.0588	0.0603	0.127	0.1519	0.279
6.8	0.0057	0.0139	0.0099	0.0243	0.0252	0.0555	0.0476	0.118	0.1382	0.257
7.0	0.0052	0.0136	0.0089	0.0235	0.0225	0.0514	0.0403	0.108	0.1302	0.237

## 4.5.2. Численные результаты

Для оценки вычислительной сложности была выполнена программная реализация на языке С++ итеративного интерполяционного алгоритма и предложенного гибридного алгоритма, были произведены замеры времени интерполяции для различных параметров, результаты которых представлены в табл. 4.1 и 4.2. Преобразование перекодирования применялось как для предложенного алгоритма (НА), так и для итеративного интерполяционного алгоритма (ПА). Видно, что предложенный подход демонстрирует уменьшение сложности более чем в два раза, и выигрыш возрастает с увеличением отношения сигнал/шум.

## 4.6. Комбинаторно-алгебраическое декодирование

### 4.6.1. Описание алгоритма

Метод Кёттера-Варди способен найти все кодовые слова  $(c_0, \dots, c_{n-1})$ , для которых

$$\sum_{i=0}^{n-1} M_{i,c_i} \geq \sqrt{2(k-1) \sum_{i=0}^{n-1} \sum_{j=0}^{q-1} \frac{M_{i,j}(M_{i,j}+1)}{2}}. \quad (4.8)$$

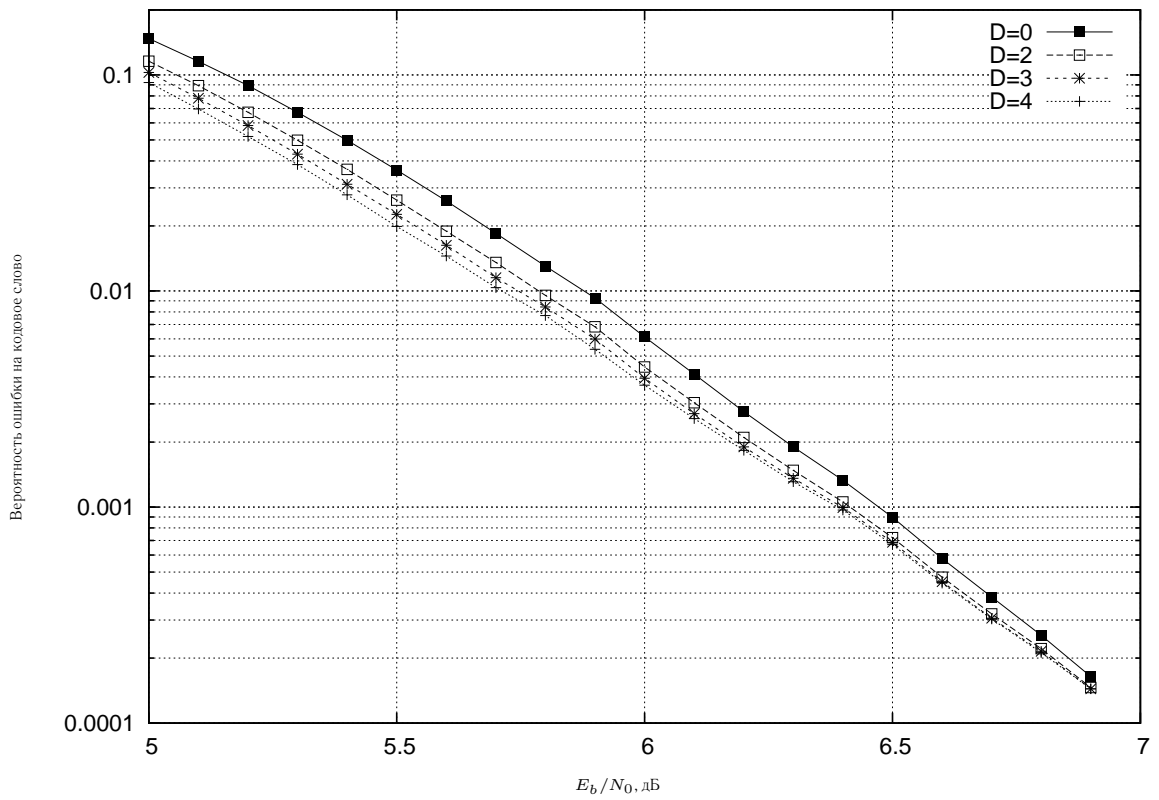


Рис. 4.5. Вероятность ошибки декодирования (63, 55) кода Рида-Соломона

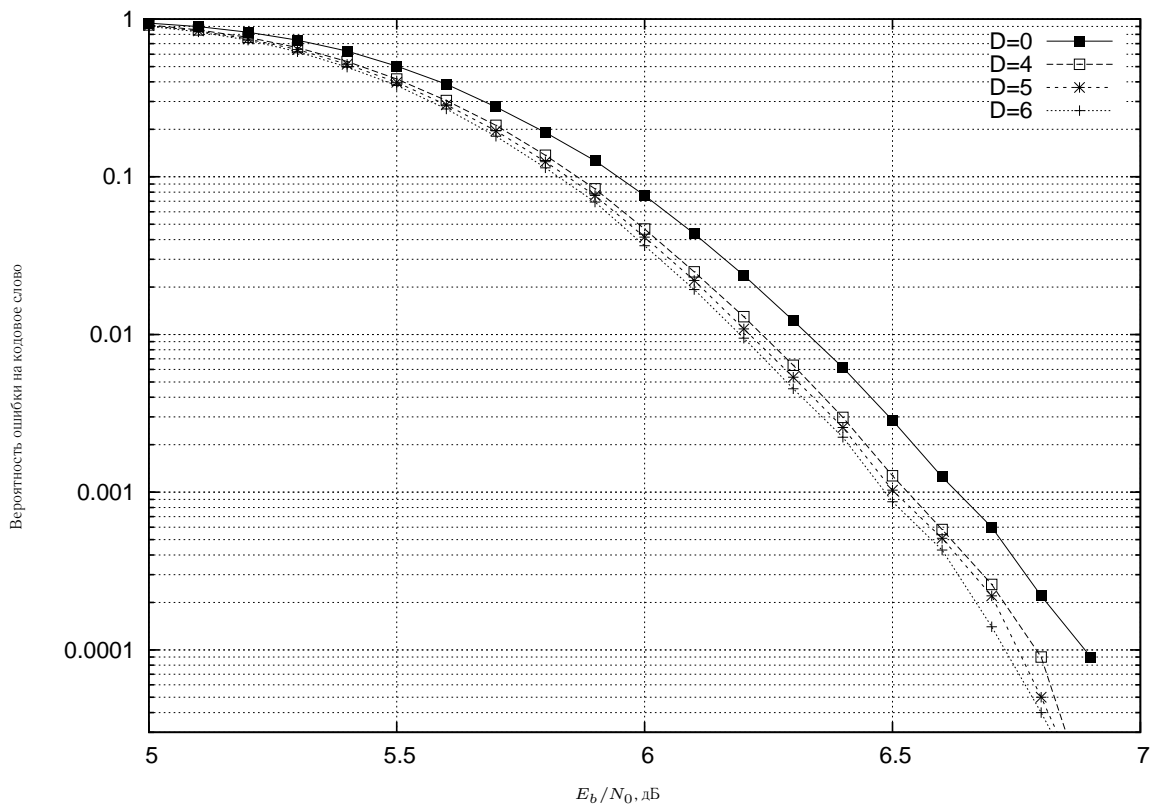


Рис. 4.6. Вероятность ошибки декодирования (255, 239) кода Рида-Соломона

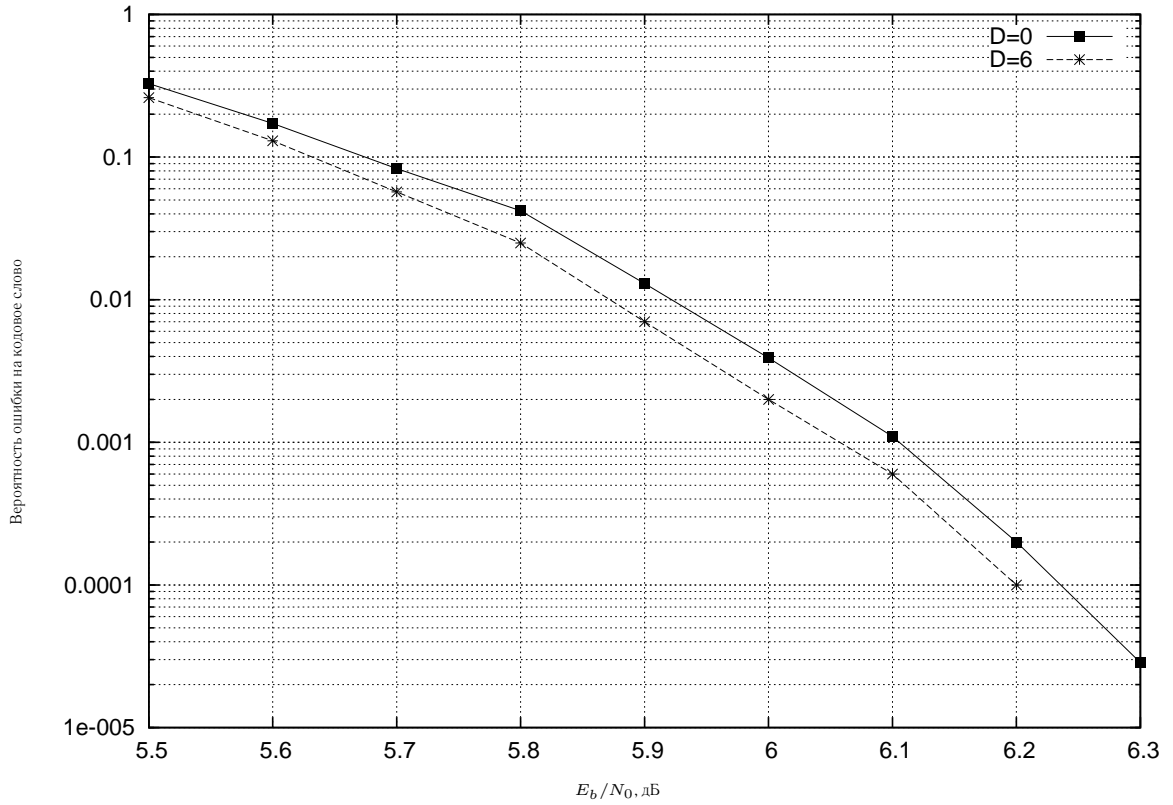


Рис. 4.7. Вероятность ошибки декодирования (511, 467) кода Рида-Соломона

Разделим принятые символы на две группы: наименее надежные символы — символы, которым соответствуют наименьшие кратности корней, и надежные символы — остальные. Пусть  $N$  — множество индексов надежных символов, а  $\hat{N}$  — наименее надежных; и пусть каждому наименее надежному символу соответствует  $\lambda > 1$  значений  $y_j$ , и кратности таких интерполяционных точек равны  $\mu$ . Тогда неравенство (4.8) преобразуется следующим образом

$$\sum_{i \in N} M_{i,c_i} + |\hat{N}| \mu \geq \sqrt{2(k-1) \left( \sum_{i \in N} \sum_{j=0}^{q-1} \frac{M_{i,j}(M_{i,j}+1)}{2} + |\hat{N}| \lambda \frac{\mu(\mu+1)}{2} \right)}.$$

Количество наименее надежных символов во много раз меньше количества надежных символов, исходя из чего, мы можем оценить правую часть неравенства как  $\sqrt{2(k-1) \sum_{i \in N} \sum_{j=0}^{q-1} \frac{M_{i,j}(M_{i,j}+1)}{2}} + \sqrt{(k-1)/8} |\hat{N}| \lambda \mu(\mu+1)$ . Отсюда видно, что с увеличением числа малонадежных символов правая часть неравенства (4.8) растет быстрее, чем левая. Это приводит к снижению числа кодовых слов, которые могут быть найдены с помощью метода Кэттера-Варди, то есть умень-

шению корректирующей способности. Таким образом, возникает необходимость разработки иных способов обработки малонадежных символов.

Предлагаемый подход состоит в объединении метода Чейза и метода Кёттера-Варди, что позволяет произвести отдельную обработку надежных и малонадежных символов. Более конкретно, предлагается произвести многократное декодирование принятой последовательности с использованием метода Кёттера-Варди, учитывая при каждой попытке различные наборы значений наименее надежных символов. Такой метод позволяет найти все кодовые слова, удовлетворяющие неравенству

$$\sum_{i \in N} M_{i,c_i} \geq \sqrt{2(k-1) \sum_{i \in N} \sum_{j=0}^{q-1} \frac{M_{i,j}(M_{i,j}+1)}{2}}.$$

Таким образом, мы увеличили корректирующую способность по сравнению с предложенным гибридным алгоритмом, поскольку  $|\hat{N}| \mu < \sqrt{(k-1)/8} |\hat{N}| \lambda \mu (\mu + 1)$ . При этом, однако, возникает проблема эффективной реализации предложенного метода декодирования. Она может быть решена путем обобщения алгоритма, описанного в [16], а также вышеописанного интерполяционного алгоритма

Предлагаемый подход состоит в упорядочении набора возможных значений ненадежных символов в виде дерева. Корень этого дерева соответствует набору высоконадежных принятых символов. Они обрабатываются с помощью классического метода Кёттера-Варди и вышеописанного алгоритма *HybridInterpolation*. Результатом его работы является базис Грёбнера модуля интерполяционных многочленов. Далее при обходе дерева могут быть рекурсивно сформированы базисы Грёбнера подмодулей этого модуля, соответствующие различным значениям ненадежных символов. Данный подход обеспечивает многократное использование результатов промежуточных вычислений, позволяя таким образом существенно снизить вычислительную сложность декодирования.

Не ограничивая общности, будем считать, что первые  $D$  символов принятого вектора являются наименее надежными.

Пусть  $M$  – матрица кратностей корней, формируемая на первом этапе метода Кёттера-Варди. Построим следующее дерево матриц кратностей корней:

- В корне данного дерева (0-й уровень) находится матрица кратностей корней  $\hat{M}^{(0)}$ , которая получается из  $M$  путем обнуления кратностей, соответствующих выбранным  $D$  точкам  $x_i$ .
- Из вершины, находящейся на  $(i - 1)$ -ом уровне, выходит  $|V_{i-1}|$  ветвей, где  $V_{i-1} = \{(x_i, y_j) | j = 0, \dots, |F| - 1, M_{i,j} > 0\}$ . Каждая ветвь ведет к вершине, которой соответствует своя точка  $(x_i, y_j)$  и  $\hat{M}^{(i)}$ , равная  $\hat{M}^{(i-1)}$  за исключением элемента  $\hat{M}_{i,j}^{(i)} = M_{i,j}$ .
- На листьях дерева (уровень  $D$ ) находятся матрицы кратностей корней, соответствующие различным пробным векторам.

Для полученных матриц необходимо построить базисы Грёбнера модулей  $\left\{ Q(x, y) = \sum_{i=0}^{\rho} y^i q_i(x) \mid Q^{[j_1, j_2]}(x_i, y_j) = 0, j_1 + j_2 < \hat{M}_{i,j}^{(D)} \right\}$ , имеющих корни  $(x_i, y_j)$  кратности  $\hat{M}_{i,j}^{(D)}$ . Для этого будем действовать следующим образом. С помощью послойного алгоритма построим базис модуля для матрицы  $\hat{M}^{(0)}$ . Базис модуля для матрицы из  $i$ -го уровня будем формировать из соответствующего базиса модуля  $(i - 1)$ -го слоя путем обработки с помощью итеративного интерполяционного алгоритма, описанного в разделе 1.5.2.1. В результате такой обработки у полиномов модуля появляется требуемый корень  $(x_i, y_j)$  кратности  $M_{i,j}$ . Описанная процедура позволяет многократно использовать промежуточные результаты вычислений, что способствует снижению сложности декодера.

В результате вышеописанных действий формируются базисы Грёбнера модулей многочленов, имеющих корни, кратности которых задаются матрицами кратностей корней, расположенными на листьях дерева. Далее из каждого найденного базиса выбирается полином с наименьшей  $(1, -1)$ -взвешенной степенью, вычисляются функциональные корни данного полинома и строятся соответствующие им кодовые слова. Результатом работы алгоритма является список

HYBRIDCHASE( $M, D$ )

- 1 Let  $M = M^N + M^{\hat{N}}, \quad \forall \max_{0 \leq j \leq |\mathbb{F}_q|-1} M_{i,j}^N \geq \max_{0 \leq j \leq |\mathbb{F}_q|-1} M_{i,j}^{\hat{N}}, \left| \left\{ i | \text{wt}(M_{i,-}^{\hat{N}}) > 0 \right\} \right| = D$
- 2  $\mathcal{B} \leftarrow \text{HYBRIDINTERPOLATION}(M^N)$
- 3  $List \leftarrow null$
- 4 CHASETREE( $\mathcal{B}, M^{\hat{N}}, List$ )
- 5 **return**  $List$

Рис. 4.8. Комбинаторно-алгебраический алгоритм декодирования

найденных кодовых слов.

Предлагаемый алгоритм представлен на Рис. 4.8 и 4.9. Функция *ChaseTree* производит рекурсивную обработку наименее надежных символов: на каждой итерации для фиксированного индекса  $i$  по очереди добавляются корни  $(x_i, y_j)$  кратности  $W_{i,j}$  к полиномам базиса  $\mathcal{B}$ . Если не существует индекса  $i$ , для которого  $W_{i,j} > 0$ , то в качестве полинома с наименьшей взвешенной степенью в базисе  $\mathcal{B}$  выбирается полином  $V$ . Кодовые слова, задаваемые соответствующими функциональными корнями, полученными в результате факторизации, добавляются в итоговый список кодовых слов. Данный список является результатом работы алгоритма. Далее необходимо воспользоваться информацией о надежности принятых символов для выбора наиболее вероятного кодового слова.

#### 4.6.2. Численные результаты

На Рис. 4.5 представлен график зависимости вероятности ошибки декодирования от отношения сигнал/шум для (63, 55)-кода над  $\mathbb{F}_{64}$ . Кривые соответствуют реализации предлагаемого комбинаторно-алгебраического алгоритма для максимальной кратности корней  $\lambda = 4$  при различных значениях параметра  $D$ . Случай  $D = 0$  соответствует классическому методу Кёттера-Варди. На Рис. 4.6 и Рис. 4.7 представлены аналогичные графики для (255, 239)-кода и (511, 467)-кода, соответственно. Из полученных результатов видно, что увеличение этого параметра приводит к росту корректирующей способности декодера.

```

CHASETREE( $\mathcal{B}, W, List, \widehat{D}$ )
1   $i \leftarrow \min_{W_{i,j} > 0} \widehat{i}$ 
2  if  $i = null$ 
3    then  $V \leftarrow \min_{0 \leq j \leq \rho} B_j$ 
4         $\widehat{List} \leftarrow \text{FACTORIZE}(V)$ 
5         $List \leftarrow List, \widehat{List}$ 
6    return  $List$ 
7   $\widehat{\mathcal{B}} \leftarrow \mathcal{B}$ 
8  for  $j : W_{i,j} > 0$ 
9    do for  $j_1, j_2 : j_1 + j_2 + 1 = W_{i,j}$ 
10   do  $\rho \leftarrow |\widehat{\mathcal{B}}| - 1$ 
11        $\sigma_t \leftarrow \text{HASSE}(\widehat{B}_t, x_i, y_j, j_1, j_2, \emptyset), t = 0, \dots, \rho$ 
12        $t_0 \leftarrow \arg \min_{t: \sigma_t \neq 0} \text{wdeg}_{(1,-1)} \text{LT } Q_t(x, z)$ 
13       if  $t_0 = \rho$ 
14         then  $\widehat{B}_{\rho+1} \leftarrow \widehat{B}_\rho \phi(x)z$ 
15              $\sigma_{\rho+1} \leftarrow \text{HASSE}(\widehat{B}_{\rho+1}, x_i, y_j, j_1, j_2, \emptyset)$ 
16              $\rho \leftarrow \rho + 1$ 
17              $\widehat{B}_t(x, z) \leftarrow \widehat{B}_t(x, z) - \frac{\sigma_t}{\sigma_{t_0}} \widehat{B}_{t_0}(x, z), t \neq t_0$ 
18              $\widehat{B}_{t_0}(x, z) \leftarrow \widehat{B}_{t_0}(x, z)(x - x_i)$ 
19              $\widehat{W} \leftarrow W, \widehat{W}_{i,j} \leftarrow 0, \forall j = 0, \dots, |\mathbb{F}_q| - 1$ 
20             CHASETREE( $\widehat{\mathcal{B}}, \widehat{W}, List$ )
21          $\widehat{\mathcal{B}} \leftarrow \mathcal{B}$ 
22 return  $List$ 

```

Рис. 4.9. Рекурсивный учет малонадежных символов



Таблица 4.3. Время интерполяции при различных  $D$ , с

$E_b/N_0$	(63, 55)-код		(255, 239)-код		(511, 467)-код	
	$D = 0$	$D = 4$	$D = 0$	$D = 6$	$D = 0$	$D = 6$
5.0	0.0008	0.0011	0.0109	0.0176	0.0513	0.0700
5.5	0.0007	0.0007	0.0104	0.0159	0.0493	0.0654
6.0	0.0005	0.0005	0.0099	0.0137	0.0487	0.0593
6.5	0.0004	0.0003	0.0092	0.0095	0.0469	0.0462
7.0	0.0003	0.0002	0.0079	0.0064	0.0455	0.0409

Оценка вычислительной сложности приведена в Таблице 4.3. Видно, что при больших отношениях сигнал/шум предлагаемый комбинаторно-алгебраический алгоритм обеспечивает выигрыш по сложности, наряду с выигрышем по корректирующей способности.

#### 4.7. Выводы по разделу

В данном разделе предложен новый комбинаторно-алгебраический алгоритм декодирования, построенный на базе метода Кёттера-Варди. Для реализации интерполяционного шага метода Кёттера-Варди разработан эффективный алгоритм двумерной интерполяции, в основе которого лежит идея двоичного метода возведения в степень. Результаты моделирования свидетельствуют о том, что предлагаемый алгоритм двумерной интерполяции обеспечивает двукратное снижение сложности по сравнению с итеративным интерполяционным алгоритмом. Корректирующая способность предлагаемого комбинаторно-алгебраического алгоритма декодирования превосходит корректирующую способность метода Кёттера-Варди благодаря использованию метода Чейза.

# Кодирование для систем хранения данных

Существует множество практических приложений, использующих коды исправляющие ошибки и требующих применения систематического кодирования.

В данном разделе предлагается алгоритм систематического кодирования, предназначенный для полярных кодов с произвольным двоичным ядром поляризации. Предлагаемый алгоритм основан на методах систематического кодирования полярных кодов с ядром Арикана (см. раздел 1.2) и обобщенных каскадных кодов (ОКК) [1]. При использовании быстрого алгоритма векторно-матричного умножения достигается дополнительное снижение сложности по сравнению с методом систематического кодирования Арикана.

### 5.1. Систематическое кодирование

Идея предлагаемого алгоритма заключается в сведении задачи систематического кодирования полярным кодом длины  $l^m$  с  $l \times l$  ядром поляризации  $M$  к  $l$  задачам систематического кодирования полярным кодом длины  $l^{m-1}$  и умножения вектора на матрицу  $M^{-1}$ . Алгоритм предполагает, что ядро  $M$  является нижнетреугольной матрицей с единичной диагональю. В дальнейшем будет показано, что это предположение не влияет на корректирующую способность, достижимую используемыми полярными кодами. Пусть множества  $\mathcal{N}$  и  $\mathcal{F}$  — множества индексов информационных и замороженных символов, соответственно. Воспользуемся представлением операции кодирования<sup>1</sup>  $c_0^{n-1} = u_0^{n-1} M^{\otimes m}$  в виде  $(m + 1)$ -слойного преобразования  $\mathcal{U}$ , где слой  $\mathcal{U}^{(0)}$  соответствует входным символам  $u_0^{n-1}$ , слои  $1, \dots, m - 1$  соответствуют результатам промежуточных вычислений  $\mathcal{U}^{(1)}, \dots, \mathcal{U}^{(m-1)}$ , в то время как слой  $\mathcal{U}^{(m)}$  содержит результат выполнения

---

<sup>1</sup> Для простоты изложения перестановочная матрица  $\Lambda^{(l,m)}$  здесь не учитывается.

операции кодирования — кодовое слово  $c_0^{n-1}$ . Каждый слой  $\mathcal{U}^{(j)}$  можно рассматривать как  $n/l^q \times l^q$  матрицу  $U^{(j,q)}$ , где  $U_{h,i}^{(j,q)} = u_{hl^q+i}^j$ ,  $0 \leq h < n/l^q$ ,  $0 \leq i < l^q$ .

Пусть  $Q_{\mathcal{B},\mathcal{D}}$  — подматрица матрицы  $Q$ , состоящая из строк и столбцов с индексами из множеств  $\mathcal{B}$  и  $\mathcal{D}$ , соответственно. Матрица  $Q_{\mathcal{B},-}$  — подматрица матрицы  $Q$ , состоящая из строк с индексами из множества  $\mathcal{B}$ .

**Лемма 1.** *Произвольное ядро поляризации  $Q$  может быть преобразовано в нижнетреугольное ядро  $M$  с единичной диагональю такое, что смежные классы  $\tilde{\mathcal{C}}^{(M,i)} = M_{i,-} + \mathcal{C}^{(M,i+1)}$  и  $\tilde{\mathcal{C}}^{(Q,i)} = Q_{i,-} + \mathcal{C}^{(Q,i+1)}$  эквивалентны, то есть существует перестановка позиций, преобразующая кодовые слова смежного класса  $\tilde{\mathcal{C}}^{(M,i)}$  в  $\tilde{\mathcal{C}}^{(Q,i)}$ . Для всех  $0 \leq i < l$  используется одна и та же перестановка. Коды  $\mathcal{C}^{(M,i)}$  и  $\mathcal{C}^{(Q,i)}$  порождаются матрицами  $M_{i..l-1,-}$  и  $Q_{i..l-1,-}$ , соответственно.*

*Доказательство.* Заметим, что добавление строк  $i+1, \dots, l-1$  к строке  $i$  матрицы  $Q$  не ведет к изменению смежного класса  $\tilde{\mathcal{C}}^{(Q,i)}$ . Действительно,  $Q_{i,-} + \sigma + \mathcal{C}^{(Q,i+1)}$ , где  $\sigma = \sum_{h=i+1}^{l-1} \alpha_h Q_{h,-}$ , является тем же самым смежным классом, что и  $Q_{i,-} + \mathcal{C}^{(Q,i+1)}$ , поскольку  $\sigma \in \mathcal{C}^{(Q,i+1)}$ . Следовательно, можно воспользоваться методом Гаусса для преобразования ядра  $Q$  в нижнетреугольное ядро  $M$ , при этом перестановки столбцов используются для того, что переместить единицы на главную диагональ, а операции над строками используются для исключения единиц над диагональю.  $\square$

Как показано в разделе 1.1, скорость поляризации зависит от частичных расстояний ядра. Напомним, что  $i$ -е частичное расстояние  $D_i$  ядра  $Q$  равно наименьшему расстоянию Хэмминга между  $i$ -ой строкой ядра и кодовыми словами кода  $\mathcal{C}^{(Q,i+1)}$ , то есть наименьшему весу кодовых слов в смежном классе  $\tilde{\mathcal{C}}^{(Q,i)}$ ,  $0 \leq i < l-1$ .  $D_{l-1}$  равно весу Хэмминга  $Q_{l-1,-}$ . Таким образом, для любого ядра  $Q$  можно построить нижнетреугольное ядро  $M$ , обладающее такими же частичными минимальными расстояниями и скоростью поляризации.

Предлагаемый алгоритм может быть использован только для систематического кодирования тех полярных кодов, для которых информационная совокупность состоит из вложенных подмножеств. Далее будет показано, что при построении полярных кодов для СВКБПДВ канала путем замораживания подканалов с наибольшими значениями параметра Бхаттачарьи, данное ограничение выполняется.

Воспользуемся леммой 4.7 из [43], в которой рассматриваются два СВКБПДВ канала  $W : \mathcal{X} \rightarrow \mathcal{Y}$  и  $\widehat{W} : \mathcal{X} \rightarrow \widehat{\mathcal{Y}}$  такие, что  $W \preceq \widehat{W}$ , и показано, что для всех  $i$  выполняется  $W_M^{(i)} \preceq \widehat{W}_M^{(i)}$ , а следовательно и  $Z(W_M^{(i)}) \geq Z(\widehat{W}_M^{(i)})$ . Здесь  $W \preceq \widehat{W}$  служит для обозначения того, что канал  $W$  хуже канала  $\widehat{W}$ , то есть существует СВКБПДВ канал  $\widetilde{W} : \widehat{\mathcal{Y}} \rightarrow \mathcal{Y}$  такой, что  $W(y|x) = \sum_{\widehat{y} \in \widehat{\mathcal{Y}}} \widehat{W}(\widehat{y}|x) \widetilde{W}(y|\widehat{y})$ .

**Лемма 2.** Если значения параметров Бхаттачарьи  $Z(W_M^{(i)})$  битовых подканалов  $W_M^{(i)}$ , задаваемых ядром  $M$  и СВКБПДВ каналом  $W$ , убывают с ростом индекса подканала  $i = 0, \dots, l-1$ , то множество информационных символов  $\mathcal{N}$  полярного кода с матрицей поляризующего преобразования  $M^{\otimes m}$  удовлетворяет следующему условию вложенности

$$\mathcal{N}_{bl}^{(j-1)} \subset \mathcal{N}_{bl+1}^{(j-1)} \subset \dots \subset \mathcal{N}_{bl+l-1}^{(j-1)}, \quad (5.1)$$

где  $\mathcal{N}_b^{(j)} = \{i \in \{0, 1, \dots, l^j - 1\} \mid i + bl^j \in \mathcal{N}\}$ ,  $0 \leq b < l^{m-j}$ ,  $0 \leq j < m$ .

*Доказательство.* Пусть  $Z_i^{(j)}$  — значение параметра Бхаттачарьи  $i$ -ого битового подканала слоя  $j$ , задаваемом  $l \times l$  ядром  $M$ . Рекурсивно применяя лемму 4.7 из [43] для слоев  $j = m-1, \dots, 0$  получаем, что если  $Z_{bl^{j+1}}^{(j+1)} > Z_{(b+1)l^{j+1}}^{(j+1)}$ , то  $Z_{bl^{j+1}+il^j}^{(j)} > Z_{(b+1)l^{j+1}+il^j}^{(j)}$  для  $0 \leq i < l$ ,  $0 \leq b < l^{m-j-1} - 1$ . Заметим также, что  $Z_{il^j+s}^{(j)} = Z_{il^j}^{(j)}$ ,  $1 \leq s < l^j$ ,  $1 \leq j < m$ , в силу рекурсивной структуры матрицы  $M^{\otimes m}$ . Отсюда,  $Z_t^{(j)} > Z_{t+\rho l^j}^{(j)}$  для  $t = bl^{j+1} + il^j + s$ ,  $0 \leq b < l^{m-j}$ ,  $0 \leq i < l$ ,  $1 \leq \rho < l-i$ ,  $0 \leq s < l^j$ ,  $0 \leq b < l^{m-j-1}$ ,  $1 \leq j < m$ . Таким образом,  $t \in \mathcal{N}_{bl+i}^{(j)}$  предполагает  $t \in \mathcal{N}_{bl+i+\rho}^{(j)}$ ,  $0 \leq b < l^{m-j-1}$ ,  $i + \rho < l$ , поскольку незамороженные битовые подканалы должны обладать наименьшими значениями параметра Бхаттачарьи.  $\square$

### 5.1.1. Поляризующее преобразование с $m = 2$

Пусть  $M^{(\mathcal{N}_h^{(1)})}$  — приведенная к систематическому виду матрица  $M_{\mathcal{N}_h^{(1)}, -}$  так, что  $M_{-, \mathcal{N}_h^{(1)}}^{(\mathcal{N}_h^{(1)})}$  является единичной матрицей,  $0 \leq h < l$ . Такая матрица  $M^{(\mathcal{N}_h^{(1)})}$  всегда может быть построена, поскольку  $M$  — нижнетреугольная матрица единичной диагональю.

Операция несистематического кодирования 0-го слоя  $U^{(1,1)} = U^{(0,1)}M$  может быть заменена систематическим кодированием

$$U_{h, \mathcal{F}_h^{(1)}}^{(1,1)} = U_{h, \mathcal{N}_h^{(1)}}^{(0,1)} M_{-, \mathcal{F}_h^{(1)}}^{(\mathcal{N}_h^{(1)})}, \quad (5.2)$$

$$U_{h, \mathcal{N}_h^{(1)}}^{(1,1)} = U_{h, \mathcal{N}_h^{(1)}}^{(0,1)}, \quad (5.3)$$

где множество  $\mathcal{F}_b^{(j)} = \{0, \dots, l^j - 1\} \setminus \mathcal{N}_b^{(j)}$ ,  $0 \leq b < l^{m-j}$ ,  $0 \leq j < m$ .

Построим кодовое слово  $U^{(2,2)}$  систематического полярного кода с информационными позициями  $i \in \mathcal{N}_0^{(2)}$ . Такое расположение информационных позиций возможно, поскольку  $M^{\otimes m}$  является нижнетреугольной матрицей с единичной диагональю.

Значения проверочных символов кодового слова  $U^{(2,2)}$ , то есть  $U_{0, \mathcal{F}_0^{(2)}}^{(2,2)}$ , могут быть получены следующим образом. Пусть  $\Phi = (M^{-1})^T$ , тогда  $\Phi$  — верхнетреугольная матрица. Несистематическое кодирование 1-го слоя задается выражением  $(U^{(2,1)})^T = (U^{(1,1)})^T M$ , которое может быть преобразовано в  $U^{(1,1)} = \Phi U^{(2,1)}$ , следовательно  $U_{h,h}^{(1,1)} = \Phi_{h,h..l-1} U_{h..l-1,h}^{(2,1)}$ . Подставляя это выражение в (5.3), получаем систему линейных уравнений

$$\Phi_{h,h..l-1} U_{h..l-1, \mathcal{F}_h^{(1)}}^{(2,1)} = \left( \Phi_{h,h..l-1} U_{h..l-1, \mathcal{N}_h^{(1)}}^{(2,1)} \right) M_{-, \mathcal{F}_h^{(1)}}^{(\mathcal{N}_h^{(1)})},$$

где  $h = l - 1, l - 2, \dots, 0$ . Поскольку  $M$  имеет единичную диагональ,  $\Phi_{h,h..l-1} U_{h..l-1, \mathcal{F}_h^{(1)}}^{(2,1)} = U_{h, \mathcal{F}_h^{(1)}}^{(2,1)} + \Phi_{h,h+1..l-1} U_{h+1..l-1, \mathcal{F}_h^{(1)}}^{(2,1)}$ .

Таким образом, получаем следующее выражение для вычисления значений проверочных символов для заданных информационных  $U_{h, \mathcal{N}_h^{(1)}}^{(2,1)}$  для  $h = l - 1, l -$

2, \dots, 0

$$U_{h, \mathcal{F}_h^{(1)}}^{(2,1)} = \left( \Phi_{h, h..l-1} U_{h..l-1, \mathcal{N}_h^{(1)}}^{(2,1)} \right) M_{-, \mathcal{F}_h^{(1)}}^{(\mathcal{N}_h^{(1)})} - \Phi_{h, h+1..l-1} U_{h+1..l-1, \mathcal{F}_h^{(1)}}^{(2,1)}. \quad (5.4)$$

Предлагаемая процедура вычисления значений проверочных символов может рассматриваться как упрощенная версия алгоритма систематического кодирования обобщенных каскадных кодов [1] для случая нижнетреугольной порождающей матрицы внутреннего кода.

**Пример 5.** Рассмотрим систематическое кодирование полярного кода

$$(16, 9) \text{ с ядром } M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \text{ и } \mathcal{N} = \{3, 6, 7, 9, 10, 11, 13, 14, 15\}. \text{ Построим}$$

множества  $\mathcal{N}_0^{(1)} = \{3\}$ ,  $\mathcal{N}_1^{(1)} = \{2, 3\}$ ,  $\mathcal{N}_2^{(1)} = \{1, 2, 3\}$  и  $\mathcal{N}_3^{(1)} = \{1, 2, 3\}$ .

После инициализации информационных символов  $U_{0, \mathcal{N}_0^{(2)}}^{(2,2)}$  вычисляем значения проверочных символов согласно выражению (5.4):

1.  $U_{3,0}^{(2,1)} = U_{3,1..3}^{(2,1)} M_{-,0}^{(\mathcal{N}_3^{(1)})}$ ,
2.  $U_{2,0}^{(2,1)} = (U_{2,1..3}^{(2,1)} + U_{3,1..3}^{(2,1)}) M_{-,0}^{(\mathcal{N}_2^{(1)})} - U_{3,0}^{(2,1)}$ ,
3.  $U_{1,0..1}^{(2,1)} = (U_{1,2..3}^{(2,1)} + U_{3,2..3}^{(2,1)}) M_{-,0..1}^{(\mathcal{N}_1^{(1)})} - U_{3,0..1}^{(2,1)}$ ,
4.  $U_{0,0..2}^{(2,1)} = (U_{0,3}^{(2,1)} + U_{1,3}^{(2,1)} + U_{2,3}^{(2,1)} + U_{3,3}^{(2,1)}) M_{-,0..2}^{(\mathcal{N}_0^{(1)})} - U_{1,0..2}^{(2,1)} - U_{2,0..2}^{(2,1)} - U_{3,0..2}^{(2,1)}$ .

### 5.1.2. Поляризующее преобразование с произвольным $m$

Описанный выше метод систематического кодирования может быть обобщен на случай произвольного числа слоев. На Рис. 5.1 приведен алгоритм систематического кодирования для полярного кода длины  $l^m$ . Входные параметры алгоритма: номер слоя  $j$ , номер блока символов  $b$ , кодируемая последовательность  $w$ . Результатом работы алгоритма является последовательность проверочных символов.

```

SYSTEMCODE( $j, b, w$ )
1  if  $|\mathcal{N}_b^{(j)}| = 0$ 
2    then return  $(0, \dots, 0)$ 
3   $U_{b, \mathcal{N}_b^{(j)}}^{(j,j)} \leftarrow w$ 
4  if  $j = 1$ 
5    then return  $U_{b, \mathcal{N}_b^{(1)}}^{(1,1)} M_{-, \mathcal{F}_b^{(1)}}^{(\mathcal{N}_b^{(1)})}$ 
6  for  $h \leftarrow l - 1$  to  $0$ 
7    do  $\tilde{\mathcal{N}} \leftarrow \mathcal{N}_{bl+h}^{(j-1)}$ 
8       $\tilde{\mathcal{F}} \leftarrow \mathcal{F}_{bl+h}^{(j-1)}$ 
9       $v_{\tilde{\mathcal{N}}} \leftarrow \Phi_{h, h..l-1} U_{lb+h..l(b+1)-1, \tilde{\mathcal{N}}}^{(j,j-1)}$ 
10      $v_{\tilde{\mathcal{F}}} \leftarrow \text{SYSTEMCODE}(j - 1, bl + h, v_{\tilde{\mathcal{N}}})$ 
11      $U_{lb+h, \tilde{\mathcal{F}}}^{(j,j-1)} \leftarrow v_{\tilde{\mathcal{F}}} - \Phi_{h, h+1..l-1} U_{lb+h+1..l(b+1)-1, \tilde{\mathcal{F}}}^{(j,j-1)}$ 
12  return  $U_{lb+h, \tilde{\mathcal{F}}}^{(j,j-1)}$ 

```

Рис. 5.1. Систематическое кодирование

Заметим, для того, чтобы закодировать информационную последовательность  $I$  длины  $k$  полярным кодом длины  $n = l^m$ , осуществляется вызов функции *SystEncode* с параметрами  $(m, 0, I)$ .

Строка 3 соответствует инициализации информационных символов заданного блока символов текущего слоя. При построении 1-го слоя значения проверочных символов получаются непосредственно в результате умножения входной последовательности на соответствующую матрицу (строка 5). В противном случае, во входной последовательности выделяются  $l$  блоков, которые последовательно кодируются.

Строки 9 и 11 на Рис. 5.1 соответствуют умножению на строки матрицы  $\Phi$ , равные столбцам матрицы  $M^{-1}$ . Напомним, что  $M^{-1}$  является нижнетреугольной матрицей. Отсюда, при вычислении произведения вектора на  $h$ -ый столбец этой матрицы необходимо знать только значения  $l - h$  элементов умножаемого вектора, при этом, при переходе от  $h$ -го столбца к  $(h - 1)$ -му вектор необходимо дополнить только одним новым значением. Далее процесс последователь-

ного вычисления таких векторно-матричных произведений мы будем называть упорядоченным умножением. Упорядоченное умножение вектора  $s$  на нижнетреугольную матрицу  $M^{-1}$  состоит в таком вычислении  $z = sM^{-1}$ , при котором  $z_i$  вычисляется с использованием только значений  $s_i, \dots, s_{l-1}$ . Сложность данной операции может быть уменьшена за счет переиспользования данных промежуточных вычислений, полученных алгоритмом упорядоченного умножения на предыдущих итерациях. Таким образом, для решения задачи упорядоченного умножения может быть построен быстрый алгоритм, в дальнейшем такой алгоритм будет называться упорядоченным умножителем.

Пусть  $\mathcal{M}^{(s_{i+1..l-1})}(s_i)$  соответствует вызову такого алгоритма на  $i$ -ом шаге. Предположим, что на шаге  $(l - 1 - i)$  на вход упорядоченного умножителя поступило значение  $s_i$ , а на выходе получено  $z_i$ . Тогда строки 9 и 11 на Рис. 5.1 могут быть заменены  $v_\eta \leftarrow \mathcal{M}^{U_{lb+h+1..l(b+1)-1,\eta}^{(j,j-1)}} \left( U_{lb+h,\eta}^{(j,j-1)} \right)$  для всех  $\eta \in \tilde{\mathcal{N}}$ , и  $U_{lb+h,\tilde{\mathcal{F}}}^{(j,j-1)} \leftarrow v_{\tilde{\mathcal{F}}} - \mathcal{M}^{U_{lb+h+2..l(b+1)-1,\beta}^{(j,j-1)}} \left( U_{lb+h+1,\beta}^{(j,j-1)} \right)$  для всех  $\beta \in \tilde{\mathcal{F}}$ , соответственно.

Предлагаемый метод может быть также обобщен на случай укороченных полярных подкодов (см. разделы 2.1 и 2.2).

## 5.2. Быстрое умножение для ядра БЧХ

В данном разделе представлен алгоритм упорядоченного умножения, предназначенный для вычисления произведения вектора на матрицу, обратную ядру БЧХ, а также анализ сложности этого алгоритма. Задача вычисления  $z = sM^{-1}$  может быть представлена как задача поиска решения системы линейных уравнений  $s = zM$ . Если представить векторы в виде полиномов, то получаем

$$s(x) = \sum_{h=0}^{l-1} z_h + x \sum_{h=1}^{l-1} (z_h \pi_h(x)). \quad (5.5)$$

Пусть  $\eta(x) = (s(x) - \sum_{h=0}^{l-1} z_h)/x$ . Из (1.7) получаем  $\eta(x) = \sum_{t=0}^{\tau-1} g_t(x) \sum_{j=0}^{q_t-1} (z_{j+\mu_t} x^j)$ , где  $\mu_t = \sum_{i=0}^{t-1} q_i$ ,  $0 \leq t < \tau$ , и  $q_t =$



$\deg g_{t+1}(x) - \deg g_t(x)$  — количество строк в  $t$ -ом блоке матрицы  $M$  (см. Рис 1.1). Во внешней сумме выделим две части

$$\eta(x) = \sum_{h=1}^{l-1} (z_h \pi_h(x)) = \underbrace{\sum_{t=0}^{\tau/2-1} g_t(x) \sum_{j=0}^{q_t-1} (z_{j+\mu_t} x^j)}_{A(x)} + g_{\tau/2}(x) \underbrace{\left( \sum_{t=\tau/2}^{\tau-1} g'_t(x) \sum_{j=0}^{q_t-1} (z_{j+\mu_t} x^j) \right)}_{D(x)}, \quad (5.6)$$

где  $g'_t(x) = g_t(x)/g_{\tau/2}(x) = (\prod_{i=0}^t w_i(x)) / (\prod_{i=0}^{\tau/2} w_i(x)) = \prod_{i=\tau/2+1}^t w_i(x)$ ,  $t \geq \tau/2$ . В каждом из полученных подвыражений можно также выделить по две части (и продолжить рекурсию).

Напомним, что задача состоит в вычислении  $z(x)$  для заданного  $s(x)$ , то есть описанные выше вычисления должны быть выполнены в обратном порядке. Из (5.6) получаем

$$\eta(x) = A(x) + g_{\tau/2}(x)D(x). \quad (5.7)$$

Для заданных  $\eta(x)$  и  $g_{\tau/2}(x)$  полиномы  $A(x)$  и  $D(x)$  могут быть получены посредством деления  $\eta(x)$  на  $g_{\tau/2}(x)$ . Также напомним, что  $\eta(x)$  может быть получен из  $s(x)$ . Следовательно, сложность умножения на  $l \times l$  матрицу  $M^{-1}$  равна

$$N_{\theta}^{mult} = 2N_{\theta/2}^{mult} + \Delta(\theta/2),$$

где  $\Delta(\theta)$  — сложность деления полинома степени  $2\theta$  на полином степени  $\theta$ . Асимптотическая сложность деления  $\Delta(\theta)$  такая же, как сложность умножения полиномов степени  $\theta$  [82]. Предполагая, что она равна  $\Delta(\theta) = O(\theta \log^a \theta)$ , где  $a \leq 2$ , получаем следующее выражение для сложности быстрого упорядоченного умножения

$$N_{\theta}^{mult} = O(\theta \log^{1+a} \theta).$$

```

 $\mathcal{M}^{(s_{i+1..l-1})}(s_i)$ 
1  if  $i = 0$ 
2    then  $z_0 = s_0 - \sum_{h=1}^{l-1} z_h$ 
3    else
4       $\eta_{i-1}^{(0)} = s_i$ 
5       $\mathcal{G} \leftarrow \{g_1(x), \dots, g_{\tau-1}(x)\}$ 
6       $z_i \leftarrow \text{ORDMULTLEVEL}(1, i - 1, \mathcal{G})$ 
7  return  $z_i$ 

```

Рис. 5.2. Упорядоченное умножение

Рассмотрим упорядоченное умножение на  $M^{-1}$ . Предположим, что для заданных  $s_i, \dots, s_{l-1}$  и уже вычисленных  $z_{i+1}, \dots, z_{l-1}$ , необходимо найти  $z_i$ . На Рис. 5.2 и 5.3 представлены алгоритмы, реализующие данную операцию. Рис. 5.3 соответствует одному шагу рекурсии. Полином  $\eta^{(1)}(x)$  инициализируется полиномом  $\eta(x)$ , полином  $\eta^{(2)}(x)$  соответствует  $D(x)$  и  $\eta^{(3)}(x)$  соответствует  $A(x)$  из выражения (5.7). Полином  $g_0(x)$  не используется, поскольку он всегда равен 1. Необходимо заметить, что построение полинома  $\eta^{(2\lambda+1)}(x)$  начинается только тогда, когда все коэффициенты полинома  $\eta^{(2\lambda)}(x)$  уже найдены.

Описанный выше алгоритм вычисления  $z = sM^{-1}$  может быть реализован в соответствии с требованиями упорядоченного умножения. Таким образом, сложность упорядоченного умножения  $N_l^{ord,mult}$  равна  $N_l^{mult}$ , то есть

$$N_l^{ord,mult} = O(l \log^{1+a} l). \quad (5.8)$$

**Пример 6.** Приведем последовательность операций, выполняемую описанным выше алгоритмом для случая ядра БЧХ  $16 \times 16$ , представленного на Рис. 1.1. Вычисление полиномов  $\pi_i(x)$ :

- $\pi_i(x) = x^{i-1}g_0(x)$ ,  $1 \leq i \leq 4$ , для  $g_0(x) = 1$ .
- $\pi_i(x) = x^{i-5}g_1(x)$ ,  $5 \leq i \leq 8$ , для  $g_1(x) = 1 + x + x^4$ .

```

ORDMULTLEVEL( $\lambda, i, \{\phi_1^{(\lambda)}(x), \dots, \phi_{\xi-1}^{(\lambda)}(x)\}$ )
1  if  $\xi = 0$ 
2    then return  $\eta_i^{(\lambda)}$ 
3   $\nu \leftarrow \deg \phi_{\xi/2}^{(\lambda)}(x)$ 
4  if  $i \geq \nu$ 
5    then  $\eta_{i-\nu}^{(2\lambda)} \leftarrow \eta_i^{(\lambda)} - \sum_{j=0}^{\nu-1} \phi_j^{(\lambda)} \eta_{i-j}^{(2\lambda)}$ 
6          $\mathcal{G} \leftarrow \left\{ \frac{\phi_{\xi/2+1}^{(\lambda)}(x)}{\phi_{\xi/2}^{(\lambda)}(x)}, \frac{\phi_{\xi/2+2}^{(\lambda)}(x)}{\phi_{\xi/2}^{(\lambda)}(x)}, \dots, \frac{\phi_{\xi-1}^{(\lambda)}(x)}{\phi_{\xi/2}^{(\lambda)}(x)} \right\}$ 
7         return ORDMULTLEVEL( $y, 2\lambda, i - \nu, \mathcal{G}$ )
8  else  $\eta_i^{(2\lambda+1)} \leftarrow \eta_i^{(\lambda)} - \sum_{j=0}^i \phi_j^{(\lambda)} \eta_{i-j}^{(2\lambda)}$ 
9          $\mathcal{G} \leftarrow \left\{ \phi_1^{(\lambda)}(x), \phi_2^{(\lambda)}(x), \dots, \phi_{\xi/2-1}^{(\lambda)}(x) \right\}$ 
10        return ORDMULTLEVEL( $y, 2\lambda + 1, i, \mathcal{G}$ )

```

Рис. 5.3. Один шаг упорядоченного умножения

- $\pi_i(x) = x^{i-9}g_2(x)$ ,  $9 \leq i \leq 10$ , для  $g_2(x) = 1 + x^4 + x^6 + x^7 + x^8$ .
- $\pi_i(x) = x^{i-11}g_3(x)$ ,  $11 \leq i \leq 14$ , для  $g_3(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$ .
- $\pi_i(x) = g_4(x)$ ,  $i = 15$ , для  $g_4(x) = 1 + x + x^2 + x^3 + \dots + x^{14}$ .

Подставляя значения параметров в (5.5), получаем  $s(x) = \sum_{h=0}^{15} z_h + x \left( \sum_{h=1}^{15} z_h \pi_h(x) \right)$ , и  $\eta(x) = \sum_{h=1}^{15} z_h \pi_h(x)$ . Это выражение может быть далее преобразовано с помощью (5.7) в  $\eta(x) = \eta^{(1)}(x) = \eta^{(3)}(x) + g_2(x)\eta^{(2)}$ , где  $\eta^{(3)}(x) = (z_1 + z_2x + z_3x^2 + z_4x^3) + g_1(x)(z_5 + z_6x + z_7x^2 + z_8x^3)$  и  $\eta^{(2)}(x) = (z_9 + z_{10}x) + g_3(x)(z_{11} + z_{12}x + z_{13}x^2 + z_{14}x^3) + g_4(x)z_{15}$ . Аналогичные выражения для всех уровней рекурсии представлены на Рис. 5.4. Заметим, что  $g_{i+1}(x)$  всегда делится на  $g_i(x)$ , в данном случае  $g_3(x)/g_2(x) = 1 + x + x^2$  и  $g_4(x)/g_3(x) = 1 + x^3 + x^4$ .

Рассмотрим первый шаг алгоритма. Для заданного  $s_{15}$ , то есть  $\eta_{14}^{(1)}$ , вычислим  $\eta_6^{(2)}$ , затем  $\eta_4^{(4)}$ , в итоге получаем  $z_{15}$  равный  $\eta_0^{(8)}$ . Следующее

$$\eta^{(1)}(x) = (z_1 + z_2x + z_3x^2 + z_4x^3) + \phi_1^{(1)}(x)(z_5 + z_6x + z_7x^2 + z_8x^3) + \phi_2^{(1)}(x)(z_9 + z_{10}x) + \phi_3^{(1)}(x)(z_{11} + z_{12}x + z_{13}x^2 + z_{14}x^3) + \phi_4^{(1)}(x)z_{15}$$

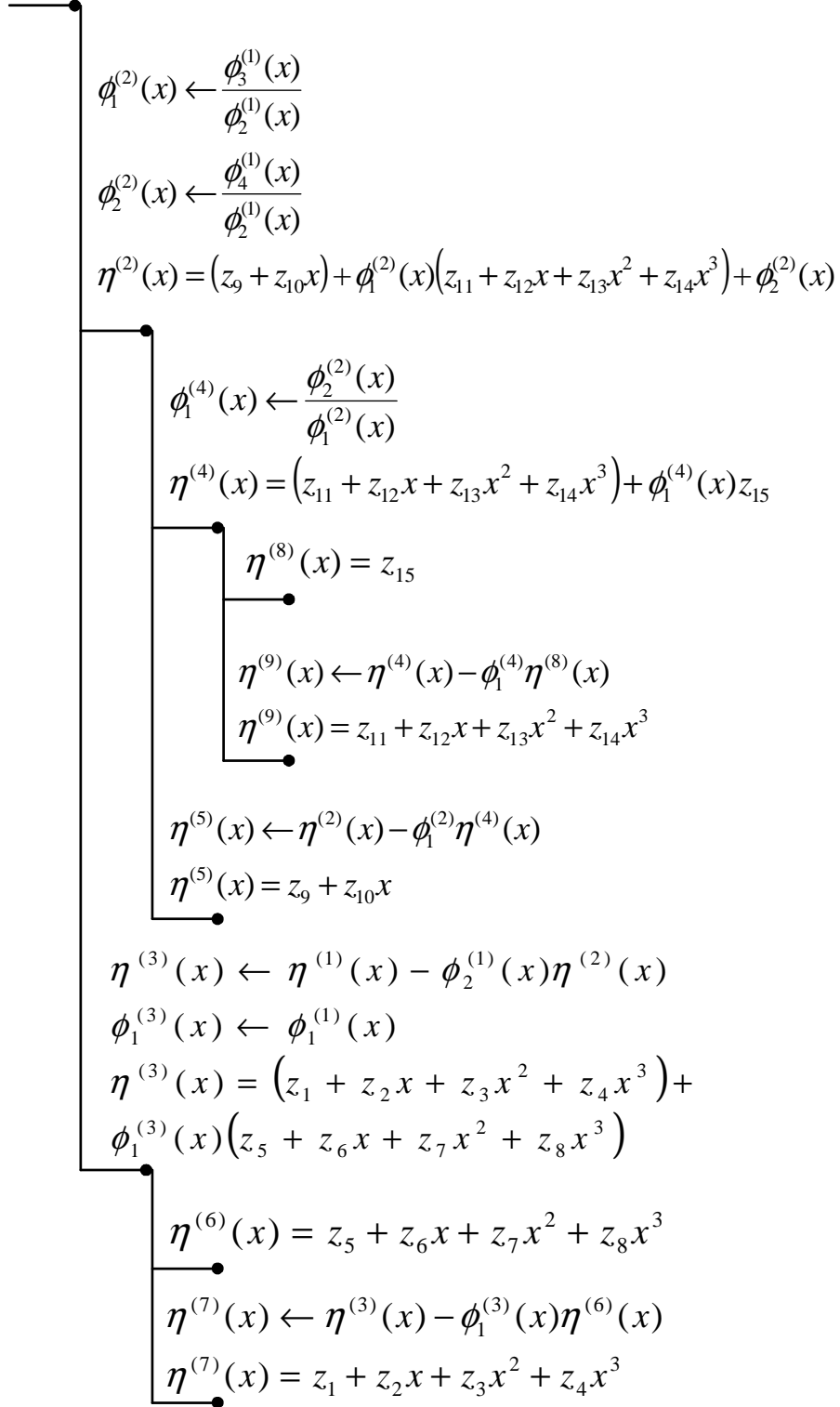


Рис. 5.4. Упорядоченное умножение на  $M^{-1}$  для ядра БЧХ  $16 \times 16$

входное значение  $s_{14}$ , то есть  $\eta_{13}^{(1)}$ . Для него вычисляем  $\eta_5^{(2)}$ , затем  $\eta_3^{(4)}$ , после чего получаем  $z_{14}$  равный  $\eta_3^{(9)}$ . Последующие вычисления выполняются аналогичным образом.

### 5.3. Анализ сложности

Пусть  $N(M)$  — сложность умножения на нижнетреугольную матрицу  $M$ ,  $N^{ord}(M)$  — сложность упорядоченного умножения на нее, и  $N^{syst}(M, m)$  — сложность предлагаемого алгоритма систематического кодирования для полярных кодов длины  $n = l^m$  с  $l \times l$  ядром  $M$ .

#### 5.3.1. Поляризующее преобразование с $m = 2$

Для оценки сложности систематического кодирования при  $m = 2$  достаточно подсчитать число сложений, необходимое для реализации (5.4). Сложность систематического кодирования выражается как

$$N^{syst}(M, 2) = N_{M^{-1}}^{syst}(M, 2) + N_M^{syst}(M, 2) + \sum_{h=0}^{l-1} |\mathcal{F}_h^{(1)}|,$$

где  $N_{M^{-1}}^{syst}(M, 2)$  — сложность умножения на столбцы матрицы  $M^{-1}$ ,  $N_M^{syst}(M, 2)$  — сложность умножения на матрицы  $M_{-\mathcal{F}_h^{(1)}}^{(\mathcal{N}_h^{(1)})}$ ,  $0 \leq h < l$ , а последнее слагаемое соответствует сложности вычитания в (5.4).

Сложность умножения на столбцы матрицы  $M^{-1}$  для  $h = l - 1, \dots, 0$  выражается как

$$N_{M^{-1}}^{syst}(M, 2, h) = |\mathcal{N}_h^{(1)}| N \left( (M^{-1})_{h..l-1, h} \right) + |\mathcal{F}_h^{(1)}| N \left( (M^{-1})_{h+1..l-1, h} \right),$$

где  $N \left( (M^{-1})_{h..l-1, h} \right)$  — сложность  $h$ -го шага упорядоченного умножения на матрицу  $M^{-1}$ , при котором для уменьшения сложности используется вложенная структура умножаемых векторов. Суммируя  $N_{M^{-1}}^{syst}(M, 2, h)$  по  $h = l - 1, \dots, 0$ , получаем  $N_{M^{-1}}^{syst}(M, 2) =$

$\sum_{h=0}^{l-1} N_{M^{-1}}^{syst}(M, 2, h) = \sum_{h=0}^{l-1} \left( (|\mathcal{N}_h^{(1)}| + |\mathcal{F}_h^{(1)}|) N \left( (M^{-1})_{h..l-1,h} \right) - |\mathcal{F}_h^{(1)}| \right) =$   
 $\sum_{h=0}^{l-1} \left( l N \left( (M^{-1})_{-,h} \right) - |\mathcal{F}_h^{(1)}| \right) = l N^{ord}(M^{-1}) - \sum_{h=0}^{l-1} |\mathcal{F}_h^{(1)}|$ . Заметим, что  
 $N \left( (M^{-1})_{h..l-1,h} \right) = N \left( (M^{-1})_{-,h} \right)$ , поскольку  $M^{-1}$  является нижнетреуголь-  
ной матрицей, и  $N \left( (M^{-1})_{h..l-1,h} \right) = N \left( (M^{-1})_{h+1..l-1,h} \right) + 1$ , поскольку все  
элементы на главной диагонали  $M^{-1}$  равны единице.

Также можно показать, что

$$N_M^{syst}(M, 2) = \sum_{h=0}^{l-1} N \left( M_{-, \mathcal{F}_h^{(1)}}^{(\mathcal{N}_h^{(1)})} \right).$$

В результате получаем следующее выражение для сложности предлагаемого алгоритма систематического кодирования

$$N^{syst}(M, 2) = l N^{ord}(M^{-1}) + \sum_{h=0}^{l-1} N \left( M_{-, \mathcal{F}_h^{(1)}}^{(\mathcal{N}_h^{(1)})} \right). \quad (5.9)$$

### 5.3.2. Поляризующее преобразование с произвольным $m$

В случае произвольного  $m$  вычисления выполняются согласно алгоритму, представленному на Рис. 5.1. Покажем, что сложность предлагаемого алгоритма систематического кодирования может быть вычислена как

$$N^{syst}(M, m) = (m-1)l^{m-1}N^{ord}(M^{-1}) + \sum_{h=0}^{l^{m-1}-1} N \left( M_{-, \mathcal{F}_h^{(1)}}^{(\mathcal{N}_h^{(1)})} \right). \quad (5.10)$$

Очевидно, что (5.10) выполняется для  $m = 2$ , поскольку совпадает с (5.9). Предположим, что выражение (5.10) справедливо для некоторого  $m$ . Тогда, подставляя выражение для сложности  $N^{syst}(M, m)$  в (5.9), получаем  $N^{syst}(M, m+1) = l^m N^{ord}(M^{-1}) + \sum_{i=0}^{l-1} \left( (m-1)l^{m-1}N^{ord}(M^{-1}) + \sum_{h=i l^{m-1}}^{(i+1)l^{m-1}-1} N \left( M_{-, \mathcal{F}_h^{(1)}}^{(\mathcal{N}_h^{(1)})} \right) \right)$ , где  $l^m N^{ord}(M^{-1})$  является сложностью кодирования на последнем слое (строки 10 и 12 на Рис. 5.1), а второе слагаемое соответствует сложности систематического кодирования  $l$  полярных кодов длины  $l^m$  (строка 11 на Рис. 5.1). Отсюда

$$\text{получаем } N^{syst}(M, m + 1) = l^m N^{ord}(M^{-1}) + l(m - 1)l^{m-1}N^{ord}(M^{-1}) + \sum_{i=0}^{l-1} \sum_{h=il+0}^{il+l^{m-1}-1} N \left( M_{-, \mathcal{F}_h^{(1)}}^{(\mathcal{N}_h^{(1)})} \right) = ml^m N^{ord}(M^{-1}) + \sum_{h=0}^{l^m-1} N \left( M_{-, \mathcal{F}_h^{(1)}}^{(\mathcal{N}_h^{(1)})} \right).$$

Заметим, что предлагаемый алгоритм систематического кодирования применим как полярным кодам, так и к произвольным двоичным ОКК с внутренними кодами, порождаемыми последними строками заданной нижнетреугольной матрицы, и систематическими внешними кодами с вложенными множествами индексов информационных символов.

### 5.3.3. Ядра Арикана и БЧХ

В случае полярного кода с ядром БЧХ подстановка выражения (5.8) для сложности упорядоченного умножения на  $l \times l$  матрицу, обратную ядру БЧХ  $B$  (см. раздел (1.1.2)), приводит к сложности систематического кодирования

$$N^{syst}(B, m) = O(n \log n \log^{1+a} l),$$

где  $n = l^m$  — длина полярного кода и  $a \leq 2$ .

Рассмотрим случай полярного кода с ядром Арикана  $A$  (см. раздел (1.1.1)). Сравним сложность предлагаемого алгоритма со сложностью алгоритма систематического кодирования Арикана, описанного в разделе 1.2, которая равна  $N^{Arikan}(m) = 1/2n \log n$ , где  $n = 2^m$ .

При  $l = 2$  получаем, что сложность предлагаемого алгоритма равна  $N^{syst}(A, m) = (m - 1)l^{m-1}N^{ord}(A^{-1}) + \sum_{h=0}^{l^{m-1}-1} N \left( A_{-, \mathcal{F}_h^{(1)}}^{(\mathcal{N}_h^{(1)})} \right) \leq (m - 1)l^{m-1} + \sum_{h=0}^{l^{m-1}-1} 1 = (m - 1)l^{m-1} + l^{m-1} = ml^{m-1} = m2^m/2 = 1/2n \log n = N^{Arikan}(m)$ , поскольку для  $l = 2$  сложность  $N^{ord}(A^{-1}) = 1$ , и  $N \left( A_{-, \mathcal{F}_h^{(1)}}^{(\mathcal{N}_h^{(1)})} \right) \in \{0, 1\}$ . Таким образом, при  $l = 2$  сложность предлагаемого алгоритма равна сложности алгоритма Арикана.

Однако, при  $m = m_0 m_1$  сложность кодирования полярного кода с ядром Арикана может быть существенно снижена посредством представления этого кода как полярного кода с ядром  $M = A^{\otimes m_0}$ , при этом  $A^{\otimes m} = M^{\otimes m_1}$ . Снижение

сложности достигается благодаря использованию оптимизированной последовательности операций для реализации умножения на  $M_{-\mathcal{F}_h}^{(\mathcal{N}_h^{(1)})}$ ,  $0 \leq h < l$ . Такой подход может быть использован только для полярных кодов, удовлетворяющих условию вложенности подмножеств индексов информационных символов (5.1), или если существует матрица перестановок  $Q$  такая, что  $QMQ^T$  — нижнетреугольная матрица, и индексы информационных символов после переупорядочения согласно  $Q$  удовлетворяют условию (5.1).

## 5.4. Применение полярных кодов в системах хранения данных

### 5.4.1. Предлагаемый метод кодирования

Большой объем используемых в современном мире цифровых данных требует построения отказоустойчивых систем для их хранения. Такие системы включают набор дисков. Отказоустойчивость системы обеспечивается за счет того, что хранимые данные кодируются с помощью кодов исправляющих ошибки. Для описания работы такой системы будем использовать модель двоичного стирающего канала с вероятностью стирания, равной вероятности отказа диска (выхода из строя диска). Следует отметить, что реальные системы хранения данных описываются намного более сложными моделями (см., например, [28]). Однако предметом данной работы являются быстрые алгоритмы кодирования данных, а не исследование надежности подобных систем. Используемая модель двоичного стирающего канала позволяет произвести выбор кодов из различных классов с близкой отказоустойчивостью.

Вероятность потери данных в системе определяется вероятностью возникновения конфигурации отказов дисков, при которой данные невозможно восстановить. Таким образом, вероятность потери данных зависит от используемого метода кодирования. В то же время, важнейшей характеристикой отказоустой-



чивых систем хранения данных является их быстродействие, которое зависит от сложности используемого алгоритма кодирования. Это препятствует использованию кодов, обладающих хорошей корректирующей способностью, но предполагающих реализацию арифметики больших конечных полей, таких как коды Рида-Соломона. Заметим, что в таких системах используется систематическое кодирование.

Предлагаемое решение состоит в использовании (укороченных) полярных кодов или полярных подкодов БЧХ с методом систематического кодирования, описанным в разделе 5.1. Описанный в разделе 2.3 метод построения полярных кодов выполняет их оптимизацию для двоичного стирающего канала и декодирования методом последовательного исключения, и может быть использован при построении кодов для отказоустойчивых систем хранения данных. Для минимизации вероятности отказа дисков в системе, нагрузка на диски должна быть сбалансирована, то есть среднее число обращений к каждому диску должно быть одинаковым.

В данном разделе представлен метод балансировки нагрузки, разработанный для системы RAID (избыточный массив независимых дисков), в которой все символы кодового слова хранятся на различных дисках. Символы кодового слова делятся на информационные и проверочные. Информационные символы могут принимать произвольные значения, в то время как значения проверочных символов равны линейным комбинациям информационных символов. При изменении значения информационного символа  $u_i$  необходимо также обновить значения проверочных символов, которые зависят от  $u_i$ , иначе хранимая последовательность символов не будет образовывать кодовое слово. Следовательно, значения проверочных символов обновляются значительно чаще, чем информационных. Это приводит к значительному дисбалансу нагрузки на диски и резкому возрастанию изношенности дисков, используемых для хранения проверочных символов.

Предлагаемый метод балансировки нагрузки предназначен для случая дво-

ичных кодов, которые могут быть представлены как обобщенные каскадные коды (ОКК) с внутренними полярными кодами с ядром Арикана (см. раздел 1.1.3). В частности, этот метод может быть использован для полярных кодов. ОКК длины  $N = vn$  задается семейством внутренних  $(v, v - i)$  кодов и внешних  $(n, k_i)$  кодов,  $0 \leq i < v$ . Если ОКК является полярным кодом длины  $N = l^m$ , построенным на базе  $l \times l$  ядра, то его внутренние длины  $v = l^s$  и внешние коды длины  $n = l^{m-s}$  также являются полярными кодами с тем же  $l \times l$  ядром. Далее описаны две составляющие предлагаемого метода балансировки нагрузки, а именно, балансировка нагрузки для групп из  $n$  дисков и балансировка нагрузки между этими  $v$  группами из  $n$  дисков.  $i$ -ая группа из  $n$  дисков соответствует последовательности символов кодового слова  $c_{i,0..n-1}$ . Для вычисления значений символов  $c_{i,k_i..n-1}$  применяется кодирование во внешнем коде  $\mathcal{C}_i$ , где  $0 \leq i < v$ . Поскольку значения  $k_i$  могут значительно отличаться, нельзя ограничиться только балансировкой внутри таких групп, то есть необходим также метод для усреднения нагрузки между этими группами дисков.

#### 5.4.2. Балансировка нагрузки для группы дисков

Предполагается, что внешние коды являются двоичными линейными блоковыми кодами, кодируемыми систематически. Рассмотрим  $(n, k)$  линейный код, порожденный матрицей  $G$ . Частота обновления  $j$ -го символа кодового слова равна

$$\lambda_j = \rho w_j, \quad 0 \leq j < n,$$

где  $\rho$  — частота обновления информационных символов и  $w_j \geq 1$  — число ненулевых элементов в  $j$ -ом столбце матрицы  $G$ . Для того, чтобы сбалансировать частоты обновления различных символов, можно использовать различные порождающие матрицы одного и того же кода.

Выполняя операции над строками исходной порождающей матрицы, можно построить  $L$  порождающих матриц  $G_i$ , содержащих  $k \times k$  единичную матрицу

$E$  на различных позициях, образующих информационную совокупность. Пусть  $w_{l,j} \geq 1$  — число ненулевых элементов в  $j$ -ом столбце матрицы  $G_l$ . В этом случае частота обновления символов выражается формулой

$$\lambda_j = \frac{\rho}{L} \sum_{l=0}^{L-1} w_{lj}, 0 \leq j < n,$$

то есть частота обновления усредняется по различным группам кодовых слов. Если выбрать матрицы  $G_l$  так, чтобы значения  $\lambda_i$  были приблизительно равны, то можно обеспечить равномерную нагрузку на диски.

**Пример 7.** Рассмотрим двоичный линейный код ( $n = 6, k = 3$ ) с порождающей матрицей  $G_0 =$

$$G_0 = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}. \text{ Можно показать, что следу-}$$

ющие матрицы порождают тот же код:  $G_1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}, G_2 =$

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}, G_3 = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}, G_4 = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}, G_5 =$$

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

Необходимо заметить, что в общем случае произвольное множество из  $k$  символов кодового слова не является информационной совокупностью, поскольку соответствующие  $k$  столбцов порождающей матрицы могут оказаться линейно зависимыми.

Для оценки качества предложенного решения используется следующая мет-

рика дисбаланса нагрузки

$$F(Q) = \sum_{j=0}^{n-1} |\lambda_j - \beta|,$$

где  $Q$  — множество информационных совокупностей и  $\beta$  — среднее  $\lambda_j$ ,  $0 \leq j < n$  по  $Q$ . Минимум функции  $F(Q)$  достигается при равномерном распределении частот обновления символов.

Построим информационные совокупности  $I^{(i)}$  для каждого внешнего кода  $\mathcal{C}_i$  с порождающей матрицей  $\mathcal{G}^{(i)}$ ,  $0 \leq i < v$ , и будем их использовать для хранения информационных символов. Однако в случае обобщенного каскадного кода информационные совокупности для внешних кодов не могут быть выбраны независимо, поскольку строки кодового слова ОКК связаны между собой. Для того, чтобы сохранялась возможность использования метода систематического кодирования, описанного в разделе 5.1, информационные совокупности должны быть вложенными. Действительно, если информационные совокупности являются вложенными  $I^{(0)} \subset I^{(1)} \subset \dots \subset I^{(v-1)}$ , то существует перестановка столбцов, применение которой для всех внешних кодов приводит к получению порождающих матриц вида  $\tilde{\mathcal{G}}^{(i)} = (E|\tilde{\mathcal{B}}^{(i)})$ , то есть  $\tilde{\mathcal{G}}^{(i)} = \mathcal{G}^{(i)}\tilde{P}$ , где  $\tilde{P}$  —  $n \times n$  матрица перестановок. Это соответствует умножению порождающей матрицы всего обобщенного каскадного кода на матрицу перестановок. К примеру, при  $v = 4$

$$\tilde{G} = G \begin{pmatrix} \tilde{P} & \mathbf{0}_{n,n} & \mathbf{0}_{n,n} & \mathbf{0}_{n,n} \\ \mathbf{0}_{n,n} & \tilde{P} & \mathbf{0}_{n,n} & \mathbf{0}_{n,n} \\ \mathbf{0}_{n,n} & \mathbf{0}_{n,n} & \tilde{P} & \mathbf{0}_{n,n} \\ \mathbf{0}_{n,n} & \mathbf{0}_{n,n} & \mathbf{0}_{n,n} & \tilde{P} \end{pmatrix} = \begin{pmatrix} (E|\tilde{\mathcal{B}}^{(0)}) & \mathbf{0}_{k_0,n} & \mathbf{0}_{k_0,n} & \mathbf{0}_{k_0,n} \\ (E|\tilde{\mathcal{B}}^{(1)}) & (E|\tilde{\mathcal{B}}^{(1)}) & \mathbf{0}_{k_1,n} & \mathbf{0}_{k_1,n} \\ (E|\tilde{\mathcal{B}}^{(2)}) & \mathbf{0}_{k_2,n} & (E|\tilde{\mathcal{B}}^{(2)}) & \mathbf{0}_{k_2,n} \\ (E|\tilde{\mathcal{B}}^{(3)}) & (E|\tilde{\mathcal{B}}^{(3)}) & (E|\tilde{\mathcal{B}}^{(3)}) & (E|\tilde{\mathcal{B}}^{(3)}) \end{pmatrix}. \quad (5.11)$$

На Рис. 5.5 представлен рандомизированный метод построения множества вложенных совокупностей  $Q^{(i)}$  для внешних кодов  $\mathcal{C}_i$ ,  $0 \leq i < v$ , минимизирующий  $\sum_{i=0}^{v-1} F(Q^{(i)})$ . Данный рандомизированный алгоритм выполняет построение  $T$  различных множеств вложенных информационных совокупностей  $Q^{(i)}$ :

```

NESTEDINFORMATIONSETSOPT( $T, L, (\mathcal{G}^{(0)}, \dots, \mathcal{G}^{(v-1)}), (\rho^{(0)}, \dots, \rho^{(v-1)})$ )
1   $Q'^{(i)} \leftarrow \emptyset, \quad 0 \leq i < v$ 
2  for  $m \leftarrow 1$  to  $T$ 
3  do  $Q^{(i)} \leftarrow \emptyset, \quad 0 \leq i < v$ 
4   $\lambda^{(i)} \leftarrow (0, \dots, 0), \quad 0 \leq i < v$ 
5  for  $l \leftarrow 1$  to  $L$ 
6  do  $I_l^{(i)} \leftarrow \emptyset, \quad 0 \leq i < v$ 
7   $q \leftarrow 1$ 
8  for  $t = 0$  to  $v - 1$ 
9  do for  $s = q$  to  $k_t$ 
10     do Поиск  $j'$  такого, что  $\mathcal{G}_{*,j'}^{(i)}$  линейно независим со всеми
         $\mathcal{G}_{*,j}^{(i)} : j \in I_l^{(i)}$ , и  $\lambda_{j'}^{(i)}$  является наибольшим возможным,
         $t \leq i < v$ 
11      $I_l^{(i)} \leftarrow I_l^{(i)} \cup \{j'\}, \quad t \leq i < v$ 
12      $q \leftarrow k_i$ 
13     Осуществление линейных операций над строками  $\mathcal{G}^{(i)}$ ,
        приводящих к получению единичной матрицы на пози-
        циях  $I_l^{(i)}$ ,  $0 \leq i < v$ 
14      $\lambda_{j'}^{(i)} \leftarrow \lambda_{j'}^{(i)} + \rho_{j'}^{(i)} \text{wt}(\mathcal{G}_{*,j'}^{(i)}), \quad 0 \leq i < v$ 
15      $Q^{(i)} \leftarrow Q^{(i)} \cup \{I_l^{(i)}\}, \quad 0 \leq i < v$ 
16      $\lambda_j^{(i)} \leftarrow \lambda_j^{(i)} / L, \quad 0 \leq i < v, 0 \leq j < n$ 
17     if  $\sum_{i=0}^{v-1} F(Q^{(i)}) < \sum_{i=0}^{v-1} F(Q'^{(i)})$ 
18     then  $Q'^{(i)} \leftarrow Q^{(i)}, 0 \leq i < v$ 
19 return  $(Q'^{(0)}, \dots, Q'^{(v-1)})$ 

```

Рис. 5.5. Построение множества информационных совокупностей

$I_l^{(0)} \subset \dots \subset I_l^{(v-1)}$ ,  $I_l^{(i)} \in Q^{(i)}$ ,  $0 \leq i < v$ , и выбирает одно из этих множеств, при котором достигается минимум  $\sum_{i=0}^{v-1} F(Q^{(i)})$ .

### 5.4.3. Балансировка нагрузки между группами дисков

Описанный выше метод для внешних кодов обеспечивает балансировку нагрузки внутри каждой строки кодового слова обобщенного каскадного ОКК. Напомним, что алгоритм систематического кодирования, предложенный в разделе 5.1, применим к произвольным двоичным ОКК с внутренними кодами  $\mathbb{C}_i$ , порождаемыми последними  $(v - i)$  строками  $v \times v$  нижнетреугольной матрицы, и систематическими внешними кодами с вложенными множествами индексов информационных символов. Последнее условие предполагает, что размерности внешних кодов упорядочены по возрастанию, то есть  $k_0 \leq k_1 \leq \dots \leq k_{v-1}$ . Следовательно, необходимо также обеспечить балансировку нагрузки внутри столбцов кодового слова ОКК, поскольку число информационных символов в  $i$ -ой строке всегда равно  $k_i$ .

Для обеспечения балансировки нагрузки внутри столбцов ОКК мы предлагаем упорядочить символы кодовых слов внутреннего кода в обратном порядке. Это приводит к появлению нового ОКК  $(N, K)$ , задаваемого тем же множеством внешних кодов  $\mathbb{C}_i$  и множеством внутренних кодов  $\mathbb{C}'_i$ ,  $0 \leq i < v$ , где  $\mathbb{C}'_i$  — код с порождающей матрицей  $Z\mathbb{G}^{(i)}$ , где

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

В дальнейшем будет использоваться обозначение  $W^Z = ZW$ . Таким образом,  $W^Z$  — матрица, симметричная матрице  $W$  относительно вертикали. При  $v = 4$  и  $\mathbb{G}^{(0)} = A^{\otimes 2}$  порождающая матрица ОКК (5.11) преобразуется следующим обра-

$$\tilde{G}' = \begin{pmatrix} \mathbf{0}_{k_0,n} & \mathbf{0}_{k_0,n} & \mathbf{0}_{k_0,n} & (E|\tilde{\mathcal{B}}^{(0)}) \\ \mathbf{0}_{k_1,n} & \mathbf{0}_{k_1,n} & (E|\tilde{\mathcal{B}}^{(1)}) & (E|\tilde{\mathcal{B}}^{(1)}) \\ \mathbf{0}_{k_2,n} & (E|\tilde{\mathcal{B}}^{(2)}) & \mathbf{0}_{k_2,n} & (E|\tilde{\mathcal{B}}^{(2)}) \\ (E|\tilde{\mathcal{B}}^{(3)}) & (E|\tilde{\mathcal{B}}^{(3)}) & (E|\tilde{\mathcal{B}}^{(3)}) & (E|\tilde{\mathcal{B}}^{(3)}) \end{pmatrix}. \quad (5.12)$$

Половина данных должна быть закодирована с использованием первоначального ОКК, порожденного матрицей (5.11), а при кодировании второй половины данных должен использоваться преобразованный ОКК, порождаемый матрицей (5.12). При этом в среднем  $i$ -ая строка кодового слова ОКК содержит  $(k_i + k_{v-i-1})/2$  информационных символов.

При использовании вышеописанного метода балансировки нагрузки возникает вопрос о его влиянии на вероятность потери данных, поскольку первоначальный ОКК  $C$  и преобразованный ОКК  $C'$  в общем случае являются различными кодами. Вероятность потери данных определяется распределением числа неисправимых конфигураций стираний (конфигураций вышедших из строя дисков). Конфигурация стирания является неисправимой в коде  $C$ , если в этом коде существуют по крайней мере два кодовых слова  $c, \tilde{c} \in C$ , которые совпадают во всех нестертых позициях, то есть при декодировании невозможно принять однозначное решение. Пусть  $\mathcal{E}(C)$  — множество неисправимых конфигураций стираний кода  $C$ . В том случае, если при кодировании данных используются коды  $C$  и  $C'$ , то конфигурация стираний является неисправимой в том случае, если она неисправима хотя бы в одном из кодов  $C$  или  $C'$ , то есть  $\mathcal{E}(C, C') = \mathcal{E}(C) \cap \mathcal{E}(C')$ . Поскольку  $\mathcal{E}(C) \subseteq \mathcal{E}(C, C')$ , то вероятность потери данных при кодировании с использованием обоих кодов  $C$  и  $C'$  не меньше вероятности потери данных при кодировании с использованием только кода  $C$ .

В рассматриваемой системе RAID предлагается использовать внутренние коды длины  $v = 2^s$ , эквивалентные кодам, порождаемым последними строками матрицы, получаемой из матрицы  $M_v = A^{\otimes s}$  в результате сортировки строк по их весу. Покажем, что в этом случае преобразованный ОКК  $C'$  совпадает с пер-

воначальным  $C$ .

Порождающую матрицу 0-го внутреннего кода можно представить как  $\mathbb{G}^{(0)} = PM_vP^T$ , при этом матрица перестановок  $P$  соответствует устойчивой сортировке строк матрицы  $\mathbb{G}^{(0)}$  в возрастающем порядке из весов. Следующая лемма показывает, что перестановка столбцов матрицы  $M_v$  в обратном порядке эквивалентна выполнению операций над строками этой матрицы.

**Лемма 3.**  $M_v^T M_v = M_v^Z$

*Доказательство.* При  $v = 2$  видно, что

$$M_2^T M_2 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix},$$

следовательно  $M_2^T M_2 = M_2^Z$ .

Для доказательства по индукции воспользуемся рекурсивной структурой матрицы Арикана

$$M_v = \begin{pmatrix} M_{v/2}^T & \mathbf{0}_{v/2, v/2} \\ M_{v/2}^T & M_{v/2}^T \end{pmatrix}.$$

Предположим, что  $M_{v/2}^T M_{v/2} = M_{v/2}^Z$  и докажем, что в этом случае  $M_v^T M_v = M_v^Z$ :

$$\begin{aligned} M_v^T M_v &= \begin{pmatrix} M_{v/2} & M_{v/2} \\ \mathbf{0}_{v/2, v/2} & M_{v/2} \end{pmatrix} \begin{pmatrix} M_{v/2} & \mathbf{0}_{v/2, v/2} \\ M_{v/2} & M_{v/2} \end{pmatrix} = \\ &= \begin{pmatrix} M_{v/2}^T M_{v/2} + M_{v/2}^T M_{v/2} & M_{v/2}^T M_{v/2} \\ M_{v/2}^T M_{v/2} & M_{v/2}^T M_{v/2} \end{pmatrix} = \begin{pmatrix} \mathbf{0}_{v/2, v/2} & M_{v/2}^T M_{v/2} \\ M_{v/2}^T M_{v/2} & M_{v/2}^T M_{v/2} \end{pmatrix} = \\ &= \begin{pmatrix} \mathbf{0}_{v/2, v/2} & M_{v/2}^Z \\ M_{v/2}^Z & M_{v/2}^Z \end{pmatrix} = \left( \begin{pmatrix} \mathbf{0}_{v/2, v/2} \\ M_{v/2} \end{pmatrix}^Z \begin{pmatrix} M_{v/2} \\ M_{v/2} \end{pmatrix}^Z \right) = M_v^Z. \end{aligned}$$

□

Пусть  $\mu_i$  —  $i$ -ая строка матрицы  $M_v$ . Заметим, что  $\text{wt}(\mu_i) = 2^{\text{wt}(i)}$ , где  $\text{wt}(i)$  — вес числа  $i$  в двоичном представлении, также видно, что  $\text{wt}(i) = s - \text{wt}(2^s -$



Таблица 5.1. Число сложений при систематическом кодировании

Ядро	$l = 4$ $m = 6$	$l = 8$ $m = 4$	$l = 16$ $m = 3$	$l = 64$ $m = 2$
$A^{\otimes(12/m)}$	20664	18995	17554	15727
$B$			22217	34528

$1 - i$ ). Тогда  $\text{wt}(\mu_{v-1-i}) = \frac{2^s}{\text{wt}(\mu_i)}$  и  $\text{wt}((PM_v)_{i+1}) = \text{wt}(\mu_{\pi(i+1)}) \geq \text{wt}(\mu_{\pi(i)})$ , то есть  $\text{wt}(\pi(i+1)) \geq \text{wt}(\pi(i))$ , где  $\pi(i)$ -ая строка матрицы  $M_v$ , которая становится  $i$ -ой строкой матрицы  $PM_v$ . Следовательно  $\text{wt}((PZM_v)_{i+1}) = \text{wt}((ZM_v)_{\pi(i+1)}) = \text{wt}(\mu_{v-1-\pi(i+1)}) = 2^{\text{wt}(v-1-\pi(i+1))} = 2^{s-\text{wt}(\pi(i+1))} \leq 2^{s-\text{wt}(\pi(i))} = 2^{\text{wt}(2^s-1-\pi(i))} = \text{wt}(\mu_{v-1-\pi(i)}) = \text{wt}((ZM_v)_{\pi(i)}) = \text{wt}((PZM_v)_i)$ . Таким образом, применение перестановки, задаваемой матрицей  $P$ , к матрице  $ZM_v$  приводит к упорядочению строк матрицы по их весу. При этом аналогичный порядок строк получается, если сначала выполнить сортировку строк матрицы  $M_v$ , и только после этого выполнить обратную перестановку столбцов, то есть  $PZ = ZP$ .

Следовательно  $(\mathbb{G}^{(1)})^Z = \mathbb{G}^{(1)}Z = PM_vP^TZ = PM_vZP^T = PM_v^T M_v P^T = \underbrace{PM_v^T P^{-1}}_W \mathbb{G}^{(1)}$ . Для матрицы перестановок  $P$  справедливо  $P^{-1} = P^T$ . Таким образом,  $W = (\mathbb{G}^{(1)})^T$  — верхнетреугольная матрица. Это предполагает, что  $(\mathbb{G}^{(i)})^Z$  также являются порождающими матрицами внутренних кодов  $\mathbb{C}_i$ . В итоге получаем, что перестановка в обратном порядке столбцов порождающих матриц внутренних не влияет на множество исправимых конфигураций стираний. Таким образом, сохраняется корректирующая способность исходного ОКК.

## 5.5. Численные результаты

Таблица 5.1 иллюстрирует сложность систематического кодирования для полярных кодов (4096, 2048). Как в случае ядра, равного ядру Арикана в  $(12/m)$ -ой степени Кронекера, так и в случае  $l \times l$  ядра БЧХ, использовались

Таблица 5.2. Число арифметических операций

	Скорость кода $k/n$	Число обновленных значений					
		1	2	4	8	16	64
EvenOdd (6,4)	0.67	5.12X	4.93X	4.56X	3.88X	1.75X	1.75X
RDP (6,4)	0.67	3.56X	3.56X	3.56X	3.68X	1.75X	1.75X
PC (6,4)	0.67	6X+6M	5.5X+5.5M	3X+2.75M	3X+2.75M	3X+2.75M	3X+2.75M
PC (11,8)	0.73	10X+6M	9X+5.5M	7X+3.5M	3.9X+1.75M	3.9X+1.75M	3.9X+1.75M
PC (19,16)	0.84	10X+6M	9X+5.5M	7X+3.5M	5.25X+2M	3.25X+0.88M	3.25X+0.875M
PC (36,32)	0.89	23X+12M	16.5X+9.5M	12X+6.75M	9.13X+3.63M	4.8X+1.4M	4.8X+1.4M
PC (72,64)	0.91	41X+17M	27X+13.5M	19.3X+10M	14.4X+5.5M	11.9X+2.8M	6.25X+0.84M
УПП (20,15)	0.75	3.96X	3.88X	3.93X	3.56X	1.73X	
УПП (48,40)	0.83	4.69X	4.57X	4.57X	4.36X	3.76X	1.6X
УПП (64,55)	0.86	4.83X	4.7X	4.69X	4.57X	4.00X	1.64X
УПП (88,76)	0.86	5.69X	5.57X	5.52X	5.28X	4.54X	3.47X
УПП (128,111)	0.87	6.95X	6.81X	6.69X	6.23X	5.31X	3.73X

быстрые алгоритмы умножения на  $M_{-\mathcal{F}_h^{(1)}}^{(\mathcal{N}_h^{(1)})}$ ,  $0 \leq h < l$ , полученные с помощью метода, предложенного в [74]. В первом случае, сложность уменьшается с ростом размерности ядра. В обоих случаях сложность предлагаемого подхода меньше сложности алгоритма Арикана. Это обусловлено наличием структуры у матрицы  $M_{-\mathcal{F}_h^{(1)}}^{(\mathcal{N}_h^{(1)})}$ , благодаря которой данные промежуточных вычислений многократно переиспользуются. Однако, в случае ядра БЧХ сложность возрастает с ростом размерности ядра, и эта сложность значительно превосходит таковую для полярных кодов с ядром Арикана.

Таблица 5.2 показывает, какое число арифметических операций (“X” — исключяющее или, “M” — умножение) приходится на один информационный символ при обновлении значений заданного числа информационных символов в различных системах RAID. Результаты приведены как для ОКК с внутренними полярными кодами с ядром Арикана и внешними укороченными полярными подкодами, так и кодов Рида-Соломона. Для кодирования последних использовался алгоритм, приведенный в [77]. В свою очередь, такие ОКК также могут рассматриваться как укороченные полярные подкоды (УПП). Видно, что предлагаемый метод кодирования для систем RAID, основанных на УПП, требует меньшего числа операций по сравнению с методом, основанным на кодах Рида-Соломона (РС). Заметим, что кодирование и декодирование укороченных полярных подкодов не требует использования умножения в расширенном поле Галуа, в отличие

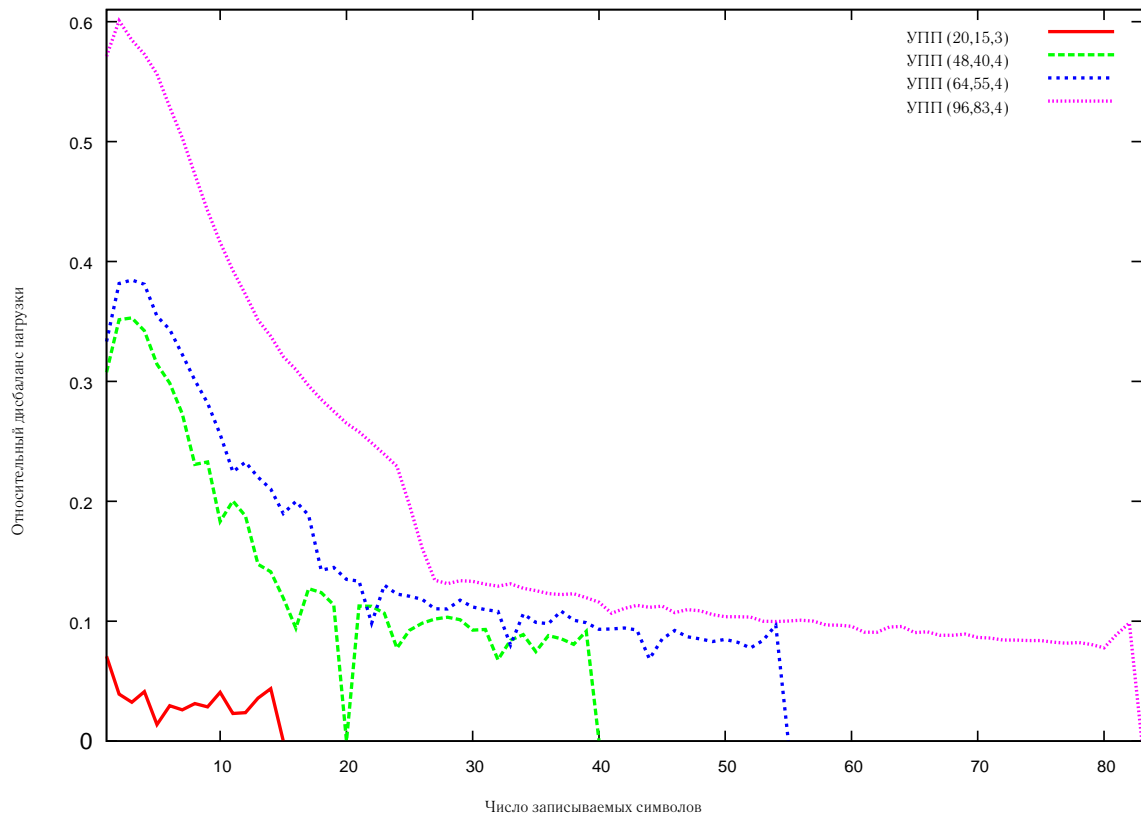


Рис. 5.6. Относительный дисбаланс нагрузки на диски при использовании предлагаемого подхода

от кодов Рида-Соломона. Однако для достижения вероятности потери данных как у кодов Рида-Соломона, кодам ОКК необходимо большее количество проверочных символов.

На Рис. 5.8 представлен график зависимости производительности операции кодирования данных от объема записываемых данных. Записываемые данные разбивались на блоки по 512 байт, каждый блок рассматривался как символ кодового слова. Представленные результаты были получены с помощью симулятора системы хранения данных, который включал в себя C++ реализацию рассматриваемых методов кодирования данных. При этом не учитывались издержки, связанные с доступом к физическим носителям данных. Эксперименты выполнялись на персональном компьютере, оснащенный процессором Intel Core i7-3930K с тактовой частотой 3.2 ГГц. Видно, что предложенный метод кодирования обеспечивает выигрыш по производительности в 1.3–2.2 раза.

На Рис. 5.6 приведен график зависимости  $F(Q)/(n\beta)$  от числа обновляемых

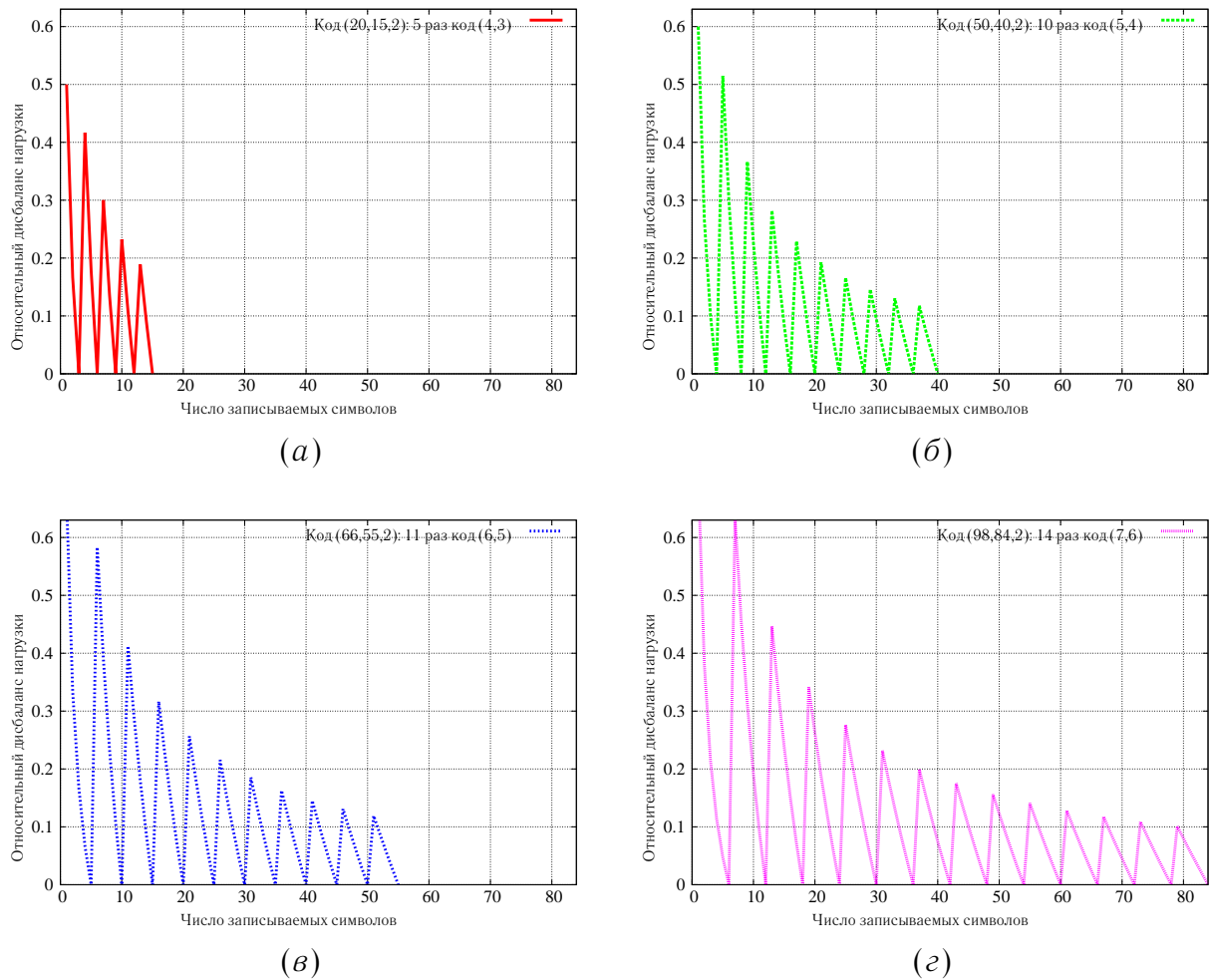


Рис. 5.7. Относительный дисбаланс нагрузки на диски при использовании RAID-4

информационных символов для предлагаемого метода кодирования данных. На Рис. 5.7 представлены аналогичные графики для кодов, построенных путем чередования кода с проверкой на четность, то есть системе, включающей несколько групп дисков, использующих RAID-4. Число дисков в группе и число групп выбраны так, чтобы параметры получаемого кода были близки к параметрам УПП, представленных на Рис. 5.6. В случае RAID-4 предполагается, что при записи последовательности из  $w$  символов осуществляется ее разбиение на  $\lfloor w/k \rfloor$  блоков из  $k$  символов и блок из  $w \bmod k$  символов, где  $k$  — размерность используемого кода с проверкой на четность. Каждый из этих блоков затем кодируется  $(k+1, k)$  кодом и осуществляется запись символов кодовых слов на диски. При таком подходе, если число  $w$  кратно размерности  $k$ , то в среднем обеспечивается равномерная нагрузка на диски, то есть  $F(Q)/(n\beta) = 0$ . В случае предлагаемого

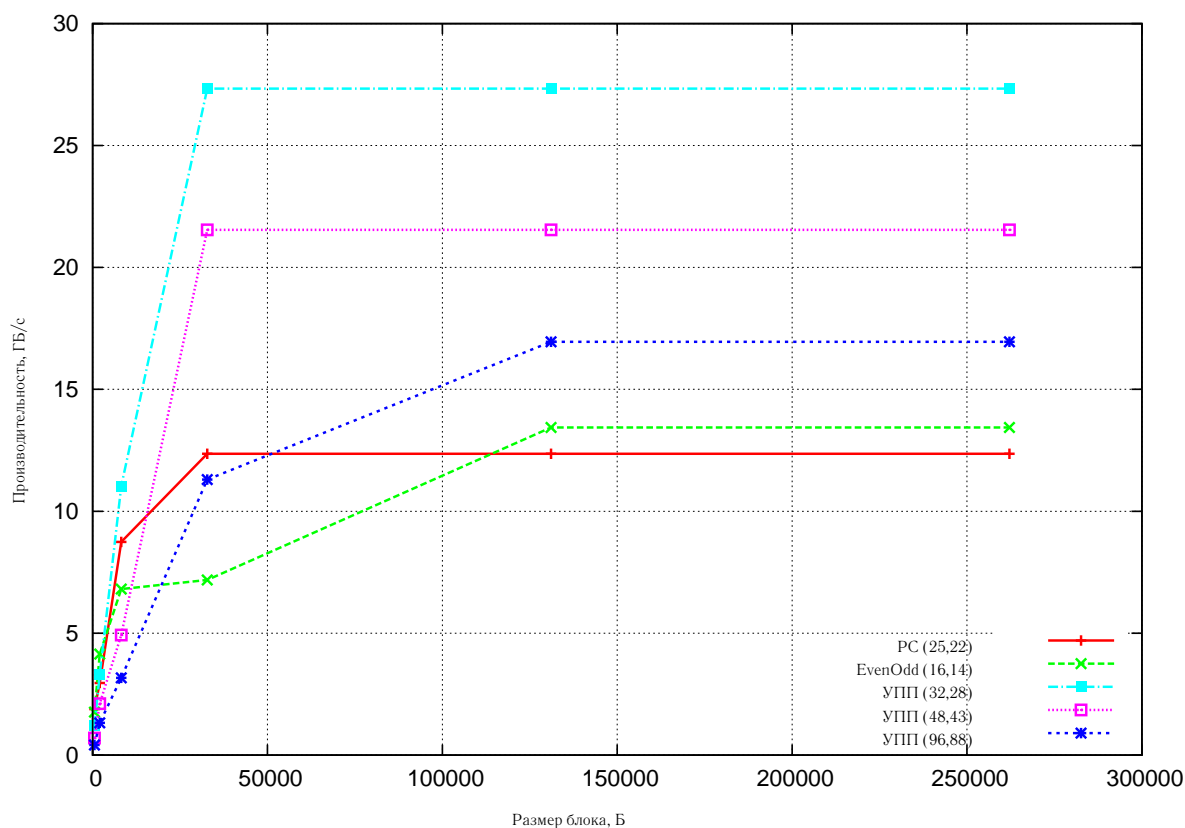


Рис. 5.8. Производительность при записи данных

метода кодирования данных, малые значения функции  $F(Q)/(n\beta)$  соответствуют тем значениям  $w$ , которым кратна размерность используемого УПП. Видно, что предлагаемый метод обеспечивает схожую степень сбалансированности нагрузки по сравнению с использованием нескольких групп дисков, каждая из которых соответствует в RAID-4. Заметим, что при этом УПП в общем случае обладают существенно большим минимальным расстоянием, в силу чего позволяют достичь меньшую вероятность потери данных.

## 5.6. Выводы по разделу

В данном разделе был представлен алгоритм систематического кодирования полярных кодов с произвольным ядром поляризации, в основе которого лежит алгоритм, предложенный в [1] для систематического кодирования обобщенных полярных кодов. В случае полярного кода с ядром Арикана асимптотическая сложность предлагаемого алгоритма равна  $O(n \log n)$  и совпадает со слож-

ностью алгоритма Арикана, где  $n$  — длина кода. В случае  $l \times l$  ядра БЧХ сложность предлагаемого алгоритма составляет  $O(n \log n \log^{1+a} l)$ , где  $a \leq 2$ .

Также было показано, что для полярных кодов с ядром Арикана длины  $n = 2^{m_0 m_1}$  предлагаемый алгоритм в сочетании с алгоритмом, выявляющим общие подвыражения при векторно-матричном умножении, позволяет получить существенный выигрыш по сложности по сравнению с алгоритмом систематического кодирования Арикана.

Было предложено использование укороченных полярных подкодов для кодирования в отказоустойчивых системах хранения данных, в которых символы кодовых слов хранятся на различных дисках. При кодировании данных используется разработанный алгоритм систематического кодирования полярными кодами. Для решения проблемы быстрого износа носителей информации, на которых хранятся значения проверочных символов, при частичном обновлении данных, был предложен метод балансировки нагрузки. Разработанный метод предназначен для кодов, которые могут рассматриваться как обобщенные каскадные коды с внутренними полярными кодами с ядром Арикана и внешними двоичными линейными кодами. Результаты моделирования свидетельствуют о том, что предлагаемый подход обеспечивает большую производительность операции кодирования по сравнению с быстрым методом кодирования кодами Рида-Соломона. При этом обеспечивается степень сбалансированности нагрузки на носители данных, близкая к случаю системы хранения данных с тем же числом дисков, организованных в несколько групп RAID-4, и избыточностью.

## Заключение

В рамках диссертационной работы был разработан набор методов построения и декодирования полярных кодов. Предложенный алгоритм декодирования был обобщен на случай коротких кодов Рида-Соломона. Кроме того, разработан быстрый алгоритм двумерной интерполяции для метода Кёттера-Варди, который позволяет производить мягкое декодирование длинных кодов Рида-Соломона. Предлагаемая конструкция полярных подкодов БЧХ в сочетании с разработанным алгоритмом последовательного декодирования обеспечивает меньшую вероятность ошибки и меньшую сложность декодирования по сравнению с существующими аналогами с практически значимыми параметрами.

Основные результаты, полученные в диссертационной работе:

1. Предложена конструкция кодов, называемых полярными подкодами БЧХ, декодирование которых может быть эффективно выполнено с помощью списочного/стекового алгоритма последовательного исключения и его аналогов.
2. Разработан алгоритм построения полярных кодов с произвольным двоичным ядром поляризации, обеспечивающий оптимальный выбор множества замороженных битовых подканалов для случая двоичного стирающего канала и декодирования методом последовательного исключения. Полученные численные результаты свидетельствуют о том, что построенные полярные коды демонстрируют хорошую корректирующую способность и в случае Гауссовского канала.
3. Предложен метод построения укороченных полярных кодов, в основе которого лежит совместная оптимизация шаблона укорочения и множества замороженных символов с целью минимизации вероятности ошибки декодирования методом последовательного исключения. Декодирование укороченного полярного кода может быть реализовано с помощью как алгорит-

ма последовательного исключения, так и его аналогов.

4. Разработан алгоритм последовательного декодирования полярных кодов с ядром Арикана. Показано, что этот алгоритм обладает существенно меньшей сложностью по сравнению с существующими списочными и стековыми алгоритмами декодирования полярных кодов. Снижение сложности достигается за счет незначительного ухудшения корректирующей способности.
5. Выполнено обобщение предлагаемого метода последовательного декодирования на случай полярных кодов с произвольным двоичным ядром. Численные результаты показывают, что предлагаемый подход позволяет выполнять декодирование полярных кодов с ядром БЧХ почти по максимуму правдоподобия.
6. На базе предложенного метода последовательного декодирования построен алгоритм декодирования коротких кодов Рида-Соломона над полем  $\mathbb{F}_{2^m}$ . Показано, что предлагаемый подход обеспечивает меньшую вероятность ошибки декодирования, чем метод Кёттера-Варди.
7. Предложен комбинаторно-алгебраический алгоритм декодирования кодов Рида-Соломона, построенный на базе метода Кёттера-Варди. Для реализации интерполяционного шага метода Кёттера-Варди разработан эффективный алгоритм двумерной интерполяции, в основе которого лежит идея двоичного метода возведения в степень. Результаты моделирования свидетельствуют о том, что предлагаемый алгоритм двумерной интерполяции обеспечивает двукратное снижение сложности по сравнению с итеративным интерполяционным алгоритмом.
8. Представлен алгоритм систематического кодирования полярных кодов с произвольным двоичным ядром поляризации. В основе этого алгоритма лежит алгоритм, предложенный в [1] для систематического кодирования



обобщенных полярных кодов. На основе данного алгоритма предложен метод кодирования укороченными полярными подкодами для отказоустойчивых систем хранения данных, включающий в себя метод балансировки нагрузки на носители информации. Результаты моделирования свидетельствуют о том, что предлагаемый подход обеспечивает большую производительность операции кодирования по сравнению с быстрым методом кодирования кодами Рида-Соломона. При этом обеспечивается степень сбалансированности нагрузки на носители данных, близкая к случаю системы, включающей несколько групп дисков, использующих RAID-4.

## Список используемых обозначений

$a_i^j$	— вектор $(a_i, a_{i+1}, \dots, a_j)$
$a_{i,\mathcal{D}}^j$	— подпоследовательность последовательности $a_i^j$ , состоящая из $a_h, h \in \mathcal{D}$
$M_{a..b}$	— матрица, состоящая из $a, \dots, b$ строк матрицы $M$
$\mathbb{F}_q$	— поле размерности $q$
$\text{LT } Q(x, y)$	— старший член полинома $Q(x, y)$
$\text{wdeg}_{(\alpha, \beta)} Q(x, y)$	— взвешенная $(\alpha, \beta)$ -взвешенная степень полинома $Q(x, y)$
$\text{xdeg } Q(x, y)$	— степень по “ $x$ ” старшего члена полинома $Q(x, y)$
$\text{ydeg } Q(x, y)$	— степень по “ $y$ ” старшего члена полинома $Q(x, y)$
$E$	— множество четных чисел
$O$	— множество нечетных чисел
$\text{supp}(v)$	— множество индексов $i$ таких, что $v_i \neq 0$
БЧХ	— (код) Боуза-Чоудхури-Хоквингема
ОКК	— обобщенный каскадный код
ЛОПП	— логарифмическое отношение правдоподобия
СВКБПДВ	— симметричный по выходу канал передачи данных без памяти с двоичным входом
КАМ	— квадратурная амплитудная модуляция
ЗРР	— зависимые по-разному распределенные (случайные величины)
НРР	— независимые по-разному распределенные (случайные величины)

## Литература

- [1] *Блох Э., Зяблов В.* Кодирование обобщенных каскадных кодов // *Проблемы передачи информации.* — 1974. — Т. 10, № 3. — С. 45–50.
- [2] *Блох Э., Зяблов В.* Обобщенные каскадные коды. — М.: Связь, 1976. — 240 с.
- [3] *Гриссер Х., Сидоренко В.* Апостериорно-вероятностное декодирование несистематических блочных кодов // *Проблемы передачи информации.* — 2002. — Т. 38, № 3. — С. 20–33.
- [4] *Зигангиров К.* Некоторые последовательные процедуры декодирования // *Проблемы передачи информации.* — 1966. — Т. 2, № 4. — С. 13–25.
- [5] *Колесник В., Мирончиков Е.* Циклические коды Рида-Маллера и их декодирование // *Проблемы передачи информации.* — 1968. — Т. 4, № 4. — С. 20–25.
- [6] *Милославская В.* Генератор двоичных полярных кодов с ядром большой размерности. — Свидетельство о государственной регистрации программы для ЭВМ 2012614815. — 2012.
- [7] *Милославская В., Трифонов П.* Гибридный алгоритм мягкого декодирования кодов Рида-Соломона // *Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление.* — 2011. — № 2. — С. 169–174.
- [8] *Заявка 2014114215 Российская Федерация, МПК G 06 F 11/00.* Способ и устройство кодирования и декодирования данных в скрученном полярном коде / Милославская В., Трифонов П.; заявитель Самсунг Электроникс Ко., Лтд.; пат. поверенный Миц А.В. — №364; заявл. 10.04.2014. — 36 с.: ил.

- [9] Трифонов П. Декодирование кодов Рида-Соломона методом последовательного исключения // *Проблемы передачи информации*. — 2014. — Т. 50, № 4.
- [10] Abedi A., Khandani A. An analytical method for approximate performance evaluation of binary linear block codes // *IEEE Transactions on Communications*. — 2004. — February. — Vol. 52, no. 2. — Pp. 228–235.
- [11] Akers S. B. Binary decision diagrams // *IEEE Transactions on Computers*. — 1978. — Vol. 27, no. 6. — Pp. 509–516.
- [12] Arikan E. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels // *IEEE Transactions on Information Theory*. — 2009. — July. — Vol. 55, no. 7. — Pp. 3051–3073.
- [13] Arikan E. Systematic polar coding // *IEEE Communications Letters*. — 2011. — August. — Vol. 15, no. 8. — Pp. 860–862.
- [14] Barg A., Zemor G. Error exponents of expander codes // *IEEE Transactions on Information Theory*. — 2002. — June. — Vol. 48, no. 6. — Pp. 1725–1729.
- [15] Becker T., Weispfenning V. Gröbner Bases. A Computational Approach to Commutative Algebra. — New York: Springer, 1993.
- [16] Bellorado J., Kavcic A. Low-complexity soft-decoding algorithms for Reed-Solomon codes—part I: An algebraic soft-in hard-out Chase decoder // *IEEE Transactions on Information Theory*. — 2010. — March. — Vol. 56, no. 3. — Pp. 945–959.
- [17] Boute R. T. The binary decision machine as a programmable controller // *EUROMICRO Newsletter*. — 1976. — Vol. 1(2). — Pp. 16–22.

- [18] *Chase D.* Code combining—a maximum-likelihood decoding approach for combining an arbitrary number of noisy packets // *IEEE Transaction on Communications*. — 1985. — May. — Vol. 33, no. 5. — Pp. 385–393.
- [19] *Chen K., Niu K., Lin J.* Improved successive cancellation decoding of polar codes // *IEEE Transactions on Communications*. — 2013. — August. — Vol. 61, no. 8. — Pp. 3100–3107.
- [20] *Ching Y.-W., Hu T.-H.* A feasible ordered statistic decoding for Reed-Solomon codes with QAM signaling // *Journal Of Chung Cheng Institute Of Technology*. — 2013. — Vol. 42, no. 2. — Pp. 23–32.
- [21] *Chung S.-Y., Richardson T. J., Urbanke R. L.* Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation // *IEEE Transactions on Information Theory*. — 2001. — February. — Vol. 47, no. 2.
- [22] *Das H., Vardy A.* Multiplicity assignments for algebraic soft-decoding of Reed-Solomon codes using the method of types // *Proceedings of IEEE International Symposium on Information Theory*. — Vol. 2. — 2009. — Pp. 1248–1252.
- [23] *Declercq D., Fossorier M. P.* Decoding algorithms for nonbinary LDPC codes over  $GF(q)$  // *IEEE Transactions on Communications*. — 2007. — April. — Vol. 55, no. 4. — Pp. 633–643.
- [24] *Delsarte P., Goethals J., MacWilliams F.* On generalized Reed-Muller codes and their relatives // *Information and control*. — 1970. — Vol. 16. — Pp. 403–442.
- [25] *Dumer I., Kabatiansky G., Tavernier C.* Soft-decision list decoding of Reed-Muller codes with linear complexity // *Proceedings of IEEE International Symposium on Information Theory*. — 2011. — Pp. 2303–2307.

- [26] 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.
- [27] *El-Khamy M., McEliece R.* Interpolation multiplicity assignment algorithms for algebraic soft decision decoding of Reed-Solomon codes // Algebraic coding theory and information theory. — 2005. — Vol. 68 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. — Pp. 99–120.
- [28] *Elerath J. G., Schindler J.* Beyond MTDDL: A closed-form RAID 6 reliability equation // *ACM Transactions on Storage*. — 2014. — March. — Vol. 10, no. 2.
- [29] *Eslami A., Pishro-Nik H.* A practical approach to polar codes // Proceedings of IEEE International Symposium on Information Theory. — 2011. — Pp. 16–20.
- [30] *Forney G.* Concatenated codes: Tech. rep.: Massachusetts Institute of Technology, 1965. — December.
- [31] *Forney G. D.* Generalized minimum distance decoding // *IEEE Transactions on Information Theory*. — 1966. — April. — Vol. 12, no. 4. — Pp. 125–131.
- [32] *Fossorier M. P., Lin S.* Soft-decision decoding of linear block codes based on ordered statistics // *IEEE Transactions on Information Theory*. — 1995. — September. — Vol. 41, no. 5. — Pp. 1379–1396.
- [33] *Guruswami V., Sudan M.* Improved decoding of Reed-Solomon and algebraic-geometric codes // *IEEE Transactions on Information Theory*. — 1999. — September. — Vol. 45, no. 6. — Pp. 1757–1767.
- [34] *Heller R.* Forced-erasure decoding and the erasure reconstruction spectra

- of group codes // *IEEE Transactions on Communication Technology*. — 1967. — Vol. 15, no. 3. — Pp. 390–397.
- [35] *Hussami N., Korada S. B., Urbanke R.* Performance of polar codes for channel and source coding // *Proceedings of IEEE International Symposium on Information Theory*. — 2009. — Pp. 1488–1492.
- [36] *Imai H., Hirakawa S.* A new multilevel coding method using error correcting codes // *IEEE Transactions on Information Theory*. — 1977. — May. — Vol. 23, no. 3. — Pp. 371–377.
- [37] *Introduction to Algorithms / T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein.* — 2 edition. — The MIT Press, 2001.
- [38] *Jiang J., Narayanan K. R.* Algebraic soft-decision decoding of Reed-Solomon codes using bit-level soft information // *IEEE Transactions on Information Theory*. — 2008. — September. — Vol. 54, no. 9. — Pp. 3907–3928.
- [39] *Kasami T., Lin S., Peterson W.* New generalizations of the Reed-Muller codes part I: Primitive codes // *IEEE Transactions on Information Theory*. — 1968. — March. — Vol. 14, no. 2. — Pp. 189–199.
- [40] *Koetter R., Ma J., Vardy A.* The re-encoding transformation in algebraic list-decoding of Reed-Solomon codes // *IEEE Transactions on Information Theory*. — 2011. — February. — Vol. 57, no. 2. — Pp. 633–647.
- [41] *Koetter R., Vardy A.* Decoding of Reed-Solomon codes for additive cost functions // *Proceedings of IEEE International Symposium on Information Theory*. — 2002. — P. 313.
- [42] *Koetter R., Vardy A.* Algebraic soft-decision decoding of Reed-Solomon codes // *IEEE Transactions on Information Theory*. — 2003. — November. — Vol. 49, no. 11. — Pp. 2809–2825.

- [43] *Korada S. B.* Polar codes for channel and source coding: Ph.D. thesis / Ecole Polytechnique Federale de Lausanne. — 2009.
- [44] *Korada S. B., Sasoglu E., Urbanke R.* Polar codes: Characterization of exponent, bounds, and constructions // *IEEE Transactions on Information Theory*. — 2010. — December. — Vol. 56, no. 12. — Pp. 6253–6264.
- [45] *Lafferty J., Vardy A.* Ordered binary decision diagrams and minimal trellises // *IEEE Transactions on Computers*. — 1999. — September. — Vol. 48, no. 9. — Pp. 971 – 986.
- [46] *Lee C. Y.* Representation of switching circuits by binary-decision programs // *Bell Systems Technical Journal*. — 1959. — Vol. 38. — Pp. 985–999.
- [47] *Lee K., O’Sullivan M. E.* An interpolation algorithm using Gröbner bases for soft-decision decoding of Reed-Solomon codes // *Proceedings of IEEE International Symposium on Information Theory*. — 2006. — Pp. 2032 – 2036.
- [48] *Li G., Fair I., Krzymien W. A.* Density evolution for nonbinary LDPC codes under gaussian approximation // *IEEE Transactions on Information Theory*. — 2009. — March. — Vol. 55, no. 3.
- [49] Low-complexity, low-memory EMS algorithm for non-binary LDPC codes / A. Voicila, D. Declercq, F. Verdier et al. // *Proceedings of IEEE International Conference on Communications*. — 2007. — June. — Pp. 671 – 676.
- [50] *Luby M.* LT codes // *Proceedings of 43rd IEEE International Symposium on Foundations of Computer Science*. — 2002. — November. — Pp. 271–280.
- [51] *Ma J., Vardy A.* A complexity reducing transformation for the Lee—O’Sullivan interpolation algorithm // *Proceedings of IEEE International Symposium on Information Theory*. — 2007. — Pp. 1986 – 1990.



- [52] *Martin P. A., Taylor D., Fossorier M. P.* Soft-input soft-output list-based decoding algorithm // *IEEE Transactions on Communications*. — 2004. — February. — Vol. 52, no. 2. — Pp. 252–262.
- [53] *Miloslavskaya V., Trifonov P.* Fast interpolation in algebraic soft decision decoding of Reed-Solomon codes // *Proceedings of IEEE R8 International Conference on Computational Technologies in Electrical and Electronics Engineering*. — 2010. — Pp. 65–69.
- [54] *Miloslavskaya V., Trifonov P.* Hybrid interpolation algorithm for algebraic soft decision decoding of Reed-Solomon codes // *Proceedings of 8th IEEE International Symposium on Wireless Communication Systems*. — 2011. — Pp. 131–135.
- [55] *Miloslavskaya V., Trifonov P.* Design of polar codes with arbitrary kernels // *Proceedings of IEEE Information Theory Workshop*. — 2012. — Pp. 119–123.
- [56] *Miloslavskaya V., Trifonov P.* Performance of binary polar codes with high-dimensional kernel // *Proceedings of International Workshop on Algebraic and Combinatorial Coding Theory*. — 2012. — Pp. 263–268.
- [57] *Miloslavskaya V., Trifonov P.* Sequential decoding of polar codes // *IEEE Communications Letters*. — 2014. — Vol. 18, no. 7. — Pp. 1127–1130.
- [58] *Miloslavskaya V., Trifonov P.* Sequential decoding of polar codes with arbitrary binary kernel // *Proceedings of IEEE Information Theory Workshop*. — 2014. — Pp. 377–381.
- [59] *Miloslavskaya V., Trifonov P.* Sequential decoding of Reed-Solomon codes // *Proceedings of International Symposium on Information Theory and Applications*. — 2014. — Pp. 424–428.

- [60] *Mori R., Tanaka T.* Performance and construction of polar codes on symmetric binary-input memoryless channels // Proceedings of IEEE International Symposium on Information Theory. — 2009.
- [61] *Nielsen R. R., Hoholdt T.* Decoding Reed-Solomon codes beyond half the minimum distance // Proceedings of the International Conference on Coding Theory and Cryptography. — Mexico: Springer-Verlag, 1998. — Pp. 221–236.
- [62] *Niu K., Chen K.* CRC-aided decoding of polar codes // *IEEE Communications Letters*. — 2012. — October. — Vol. 16, no. 10. — Pp. 1668–1671.
- [63] On the construction and decoding of concatenated polar codes / H. Mahdavi-far, M. El-Khamy, J. Lee, I. Kang // Proceedings of IEEE International Symposium on Information Theory. — 2013.
- [64] Optimal decoding of linear codes for minimizing symbol error rate / L. Bahl, J. Cocke, F. Jelinek, J. Raviv // *IEEE Transactions on Information Theory*. — 1974. — Pp. 284–287.
- [65] *Parvaresh F., Vardy A.* Multiplicity assignments for algebraic soft-decoding of Reed-Solomon codes // Proceedings of IEEE International Symposium on Information Theory. — 2003. — June. — P. 205.
- [66] *Ross A.* Computing bounds on the expected maximum of correlated normal variables // *Methodology and Computing in Applied Probability*. — 2010. — Vol. 12, no. 1. — Pp. 111–138.
- [67] *Roth R., Ruckenstein G.* Efficient decoding of Reed-Solomon codes beyond half the minimum distance // *IEEE Transactions on Information Theory*. — 2000. — Vol. 46, no. 1. — Pp. 246–257.

- [68] Roth R., Skachek V. Improved nearly-MDS expander codes // *IEEE Transactions on Information Theory*. — 2006. — August. — Vol. 52, no. 8. — Pp. 3650–3661.
- [69] Seidl M., Huber J. B. Improving successive cancellation decoding of polar codes by usage of inner block codes // *Proceedings of 6th International Symposium on Turbo Codes and Iterative Information Processing*. — 2010. — Pp. 103 – 106.
- [70] Shin D.-M., Lim S.-C., Yang K. Design of length-compatible polar codes based on the reduction of polarizing matrices // *IEEE Transactions on Communications*. — 2013. — July. — Vol. 61, no. 7. — Pp. 2593–2599.
- [71] Sorokine V., Kschischang F. A sequential decoder for linear block codes with a variable bias-term metric // *IEEE Transactions on Information Theory*. — 1998. — January. — Vol. 44, no. 1. — Pp. 410–416.
- [72] Tal I., Vardy A. List decoding of polar codes // *Proceedings of IEEE International Symposium on Information Theory*. — 2011. — Pp. 1–5.
- [73] Tal I., Vardy A. How to construct polar codes // *IEEE Transactions on Information Theory*. — 2013. — October. — Vol. 59, no. 10. — Pp. 6562–6582.
- [74] Trifonov P. Matrix-vector multiplication via erasure decoding // *Proceedings of XI International Symposium on Problems of Redundancy in Information and Control Systems*. — 2007. — Pp. 104–108.
- [75] Trifonov P. Efficient interpolation in the Guruswami-Sudan algorithm // *IEEE Transactions on Information Theory*. — 2010. — September. — Vol. 56, no. 9. — Pp. 4341–4349.
- [76] Trifonov P. Efficient design and decoding of polar codes // *IEEE Transactions*

- on Communications*. — 2012. — November. — Vol. 60, no. 11. — Pp. 3221–3227.
- [77] *Trifonov P.* Low-complexity implementation of RAID based on Reed-Solomon codes // *ACM Transactions on Storage*. — 2014. — accepted.
- [78] *Trifonov P., Miloslavskaya V.* Polar codes with dynamic frozen symbols and their decoding by directed search // *Proceedings of IEEE Information Theory Workshop*. — 2013. — September. — Pp. 1–5.
- [79] *Trifonov P., Semenov P.* Generalized concatenated codes based on polar codes // *Proceedings of IEEE International Symposium on Wireless Communication Systems*. — 2011. — Pp. 442–446.
- [80] *Valembois A., Fossorier M.* Box and match techniques applied to soft-decision decoding // *IEEE Transactions on Information Theory*. — 2004. — May. — Vol. 50, no. 5. — Pp. 796–810.
- [81] *Vitale R.* Some comparisons for gaussian processes // *Proceedings of the American Mathematical Society*. — 2000. — Vol. 128, no. 10. — Pp. 3043–3046.
- [82] *von zur Gathen J., Gerhard J.* *Modern Computer Algebra*. — Cambridge University Press, 1999.
- [83] *Wachsmann U., Fischer R. F. H., Huber J. B.* Multilevel codes: Theoretical concepts and practical design rules // *IEEE Transactions on Information Theory*. — 1999. — July. — Vol. 45, no. 5. — Pp. 1361–1391.

# Приложение А

## Акты о внедрении



УТВЕРЖДАЮ

Генеральный директор  
Центра разработок EMC в Санкт-Петербурге

В.М. Нестеров



### АКТ О ВНЕДРЕНИИ результатов диссертационной работы

Настоящим удостоверяется, что результаты диссертационной работы Милославской В.Д. «Методы построения и декодирования полярных кодов» использовались в Санкт-Петербургском центре разработки программного обеспечения корпорации EMC при разработке высокопроизводительных отказоустойчивых систем хранения данных.

Применение результатов работы позволило:

- повысить скорость кодирования данных;
- обеспечить равномерность износа носителей информации.

Руководитель проекта

Александр Алексеев  
Ведущий инженер-программист  
Центр разработок EMC в Санкт-Петербурге

EMC St. Petersburg Development Centre LLC - Russia, 199004, St. Petersburg, 36/40 Sredniy pr. V.O., Business center "Ostrov", 5<sup>th</sup> floor, phone + 7 (812) 325-4633, fax + 7 (812) 325-4607

ООО «Санкт-Петербургский Центр Разработок EMC» - Российская Федерация, 199004, Санкт-Петербург, ВО, Средний пр., 36/40, Бизнес-центр «Остров», 5-й этаж, тел. +7 (812) 325-4633, факс +7 (812) 325-4607

УТВЕРЖДАЮ

И.о. проректора по  
образовательной деятельности  
Разинкина Е.М.



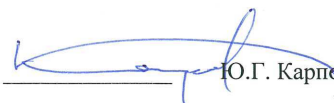
АКТ

**об использовании результатов диссертационной работы Милославской В.Д.  
«Методы построения и декодирования полярных кодов»  
в учебном процессе Санкт-Петербургского  
государственного политехнического университета**

Мы, нижеподписавшиеся, заведующий кафедрой распределенных вычислений и компьютерных сетей д.т.н., проф. Карпов Ю.Г. и зам. заведующего кафедрой по учебной работе Сениченков Ю.Б., подтверждаем, что результаты диссертационной работы Милославской В.Д. (модель полярных кодов с динамически замороженными символами и алгоритм последовательного декодирования полярных кодов) использованы в учебном процессе кафедры распределенных вычислений и компьютерных сетей в дисциплине «Помехоустойчивое кодирование» для студентов, обучающихся по направлению подготовки магистров «Фундаментальная информатика и информационные технологии».

«21» ноября 2014 года

Зав. кафедрой РВКС  
д.т.н., проф.

  
Ю.Г. Карпов

Зам. зав. кафедрой РВКС по учеб. работе  
д.т.н., проф.

  
Ю.Б. Сениченков