МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ

На правах рукописи

Акмалходжаев Акмал Илхомович

Разработка и исследование эффективных алгоритмов декодирования турбокодов в системах мобильной связи

Специальность 05.12.13 – Системы, сети и устройства телекоммуникаций

Диссертация на соискание ученой степени

кандидата технических наук

Научный руководитель

доктор технических наук,

профессор Е.А. Крук

Санкт-Петербург – 2015

ОГЛАВЛЕНИЕ

Введение

Актуальность темы. В настоящее время наблюдается активное развитие таких областей передачи информации, как мобильные сети и беспроводные технологии. Имеющиеся системы уже не удовлетворяют растущим требованиям к скорости доступа в сеть, качеству передачи видео, аудио контента и голоса. Для обеспечения требуемой мобильности пользователей и достоверности передачи данных внедряют новые решения, такие как OFDM, MIMO и др. Однако помехоустойчивое кодирование остается одним из основных способов гарантировать высокую скорость передачи информации при проектировании высокоскоростных цифровых систем связи. Различные конструкции помехоустойчивых кодов, алгоритмы их кодирования и декодирования рассматривают в своих работах такие российские ученые как В.В. Зяблов, К.Ш. Зигангиров, Е.А. Крук, Б.Д. Кудряшов и др. Важной является задача выбора наиболее подходящей конструкции для той или иной системы.

Турбокоды с уверенностью можно назвать одними из наиболее востребованных в современных сетях. Они доказали свою эффективность в стандартах WiMax и DVB-RCS. Но наибольшее распространение эти коды получили в мобильных сетях третьего и четвертого поколения UMTS и 3GPP LTE. Их широкое практическое применение обусловлено эффективным итеративным (турбо-) декодером, возможностью использования одного и того же кодера для различных длин информационных слов и гибкой схемой регулирования скорости кода.

Улучшение параметров турбокодов в разрабатываемых системах возможно за счет выбора компонентных кодов и перемежителя с одной стороны, и использования другого алгоритма декодирования с другой. Однако, принимая во внимание то, что в существующих стандартах и установленном оборудовании нет возможности изменить передающую часть, важной является задача улучшение параметров декодирования. Современные технологии и рост производительности аппаратной базы сделали возможным использование более сложных и

оригинальных декодеров для уменьшения вероятности ошибки в принимаемых данных. Учитывая тот факт, что все большую популярность получает параллельная обработка, актуальной является задача разработки распараллеливаемых алгоритмов декодирования.

В данной диссертационной работе предлагается два способа улучшения декодера турбокода: списочное декодирование и совместное декодирование турбокода и кода источника. Оба подхода учитывают специфику передачи в современных мобильных сетях, это использование CRC для обнаружения ошибок в декодированном слове и структуру информационного слова, которая включает в себя заголовки протоколов различных уровней. Поскольку передача голоса остается одним из важнейших видов трафика и важным пунктом дохода операторов связи, основное внимание при разработке алгоритма совместного декодирования кодов канала и источника уделено источнику, который представляет собой вокодер. В частности результаты моделирования приведены для турбокода стандарта 3GPP LTE и вокодеров AMR-NB и AMR-WB.

Целью работы является исследование и разработка эффективных алгоритмов декодирования турбокодов с использованием техник списочного декодирования и совместного декодирования кодов канала и источника. Применение предложенных алгоритмов в сети 3GPP LTE для улучшения качества передачи речи.

В соответствии с целью были поставлены следующие **задачи диссертационного исследования**:

1. Сравнительный анализ классических итеративных и списочных алгоритмов декодирования турбокодов.
2. Исследование методов повышения эффективности списочных декодеров турбокодов и разработка алгоритма, обеспечивающего уменьшение вероятности ошибки на пакет по сравнению с классическим турбо-декодером при задержке декодирования, приемлемой для практического использования.

3. Исследование и разработка алгоритма совместного декодирования турбокода и кода источника, который учитывает специфику передачи в современных сетях мобильной связи.
4. Исследование возможности списочного и совместного декодирования в системе сотовой связи стандарта 3GPP LTE для улучшения качества передачи речи.

Методы исследования. Для решения поставленных задач были использованы методы теории информации, методы теории вероятностей и математической статистики, методы теории кодирования, а также методы имитационного моделирования.

Научная новизна диссертационной работы заключается в следующем:

1. Разработан списочный алгоритм декодирования турбокода, который уменьшает вероятность ошибки на пакет по сравнению с классическим турбодекодером при приемлемой для практического использования задержке декодирования, чего не позволяют существующие списочные подходы.
2. Предложена оконная модификация списочного декодера турбокода, позволяющая ускорить процесс генерации списка и уменьшить задержку декодирования за счет разбиения решетки сверточного кода на сегменты и независимого вычисления априорных последовательностей в каждом сегменте.
3. Разработан алгоритм совместного декодирования турбокода и кода источника, учитывающий наличие корреляционной зависимости лишь у части бит информационного слова, что значительно снижает эффективность известных алгоритмов.
4. Оценена избыточность на выходе вокодеров AMR-NB и AMR-WB и предложен алгоритм совместного декодирования турбокода и вокодеров AMR-NB и AMR-WB, обеспечивающий улучшение качества принимаемой речи в стандарте 3GPP LTE.

Теоретическая и практическая значимость полученных результатов. Ценность данной работы заключается в том, что внедрение и использование предложенных алгоритмов при декодировании на базовых станциях и мобильных устройствах позволяет уменьшить вероятность ошибки на пакет и увеличить скорость передачи данных.

Степень достоверности результатов обеспечивается корректным использованием математического аппарата, обоснованностью используемых ограничений и допущений, использованием имитационного моделирования для оценки качества работы предложенных алгоритмов. Апробацией полученных результатов в виде докладов на научно-технических конференциях и публикациях в периодической печати.

Апробация работы. Основные результаты работы докладывались и обсуждались на следующих конференциях и симпозиумах: Научная сессия ГУАП (Санкт-Петербург, Россия, 2013); Всероссийской научной конференции по проблемам информатики «СПИСОК» (Санкт-Петербург, Россия, 2012, 2014); 14 международный симпозиум «Problems of Redundancy in Information and Control Systems» (Санкт-Петербург, Россия, 2014); Международная конференция «6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)» (Санкт-Петербург, Россия, 2014).

Внедрение результатов. Результаты работы были использованы в рамках проектов «On LTE turbo decoding performance improvement» и «VoLTE Quality Improvement and Non-intrusive Evaluation», осуществляемого ПАО «ИКТ». Кроме того, теоретические результаты работы используются в учебном процессе кафедры безопасности информационных систем ГУАП.

Публикации. Результаты, представленные в диссертационной работе, опубликованы в 9 печатных работах [1–9]. Среди них 3 работы [1, 2, 9] опубликованы в изданиях, включенных в список ВАК, и 3 работы [6–8] опубликованы в изданиях, включенных в перечень «Scopus».

Основные положения, выносимые на защиту.

1. Алгоритм списочного декодирования турбокода, уменьшающий вероятность ошибки на пакет по сравнению с классическим турбо-декодером.
2. Метод ускорения процесса генерации списка и уменьшения задержки декодирования списочного декодера турбокода.
3. Алгоритм совместного декодирования турбокода и кода источника, учитывающий наличие случайных бит в структуре информационного слова.
4. Алгоритм совместного декодирования турбокода и вокодеров AMR-NB и AMR-WB, обеспечивающий улучшение качества принимаемой речи в стандарте 3GPP LTE.

Объем и структура работы.

Диссертационная работа состоит из введения, четырех разделов, заключения и списка использованных источников (103 наименования). Диссертация содержит 126 страниц, включая 7 таблиц и 51 рисунок.

В первом разделе дается описание основных известных результатов в области турбокодов, рассмотрен классический итеративный алгоритм декодирования турбокода, который в качестве компонентного декодера использует алгоритм декодирования сверточного кода с мягким выходом. Приведены результаты моделирования для различных алгоритмов декодирования компонентных кодов в канале с АБГШ. Показано, что при турбо-декодировании алгоритм Scaled Max-Log-MAP дает производительность близкую к декодеру Log-MAP при значительно меньшей сложности реализации.

Во втором разделе исследованы списочные алгоритмы декодирования турбокодов. Рассмотрены существующие подходы, которые основаны на списочном декодировании компонентного кода, либо использующие проверочную матрицу турбокода. Показано, что с ростом длины информационного слова эффективность рассмотренных алгоритмов быстро падает, а сложность значительно возрастает. Разработан алгоритм списочного декодирования турбокода, который

уменьшает вероятность ошибки на пакет при задержке декодирования, сравнимой с обычным турбо-декодером. Предложена оконная модификация алгоритма для ускорения процесса генерации списка. Результаты моделирования приведены для турбокода стандарта 3GPP LTE и канала с АБГШ.

В разделе три рассматривается другой способ улучшения параметров декодирования - совместное декодирование кодов канала и источника. Предложен новый совместный декодер для турбокода и источника с избыточностью. При этом предполагается, что информационное слово помимо данных источника включает в себя заголовки протоколов различных уровней, что приводит к уменьшению эффективности обычных совместных декодеров. Отдельно рассмотрена избыточность вокодеров AMR-NB и AMR-WB, для которых предложен совместный декодер и продемонстрирована его эффективность.

В четвертом разделе списочный декодер турбокода применен в системе связи стандарта 3GPP LTE с целью улучшения качества передаваемой речи, что является актуальным в случае, когда совместное декодирование кодов канала и источника невозможно. Рассмотрены основные режимы кодирования голоса и для каждого из них показано получаемое от использования списочного декодера улучшение качества.

В заключении кратко перечислены основные результаты, полученные в ходе диссертационной работы.

1 Турбокоды и методы их декодирования

1.1 Вводные замечания

Впервые турбокоды были предложены в 1993 году К. Берроу, А. Главьё и П. Ситимашимой [10]. В своей работе авторы не только предложили схему кодера, но также эффективный алгоритм декодирования. Основными элементами схемы кодирования являются два рекурсивных систематических сверточных кода, связанных между собой перемежителем (рисунок 1.1). По этой причине турбокоды также называют параллельными каскадными кодами. В качестве декодера была предложена итеративная схема, ядром которой являются два декодера сверточных кодов с мягким выходом. Экспериментально было показано [10], что на длине информационного слова 65536 бит такая схема кодирования и декодирования практически достигает пропускной способности канала при реализуемой сложности декодера. В оригинальной работе [10] для декодирования сверточного кода использовался алгоритма Log-MAP [11], однако позже для уменьшения сложности декодирования турбокода были рассмотрены и другие подоптимальные декодеры с мягким выходом, такие как SOVA [12], Max-Log-MAP и Scaled Max-Log-MAP [13].

Несмотря на плохое минимальное расстояние, турбокоды обладают хорошими спектральными характеристиками и корректирующими возможностями, и практически сразу были оценены по достоинству. Много работ посвящено анализу построения турбокодов: исследовано влияние выбора перемежителя и сверточных кодов на характеристики кода [14–17], рассмотрены варианты использования трех и более компонентных кодов [18, 19]. Также значительное внимание уделено анализу итеративного декодера и способам его улучшения: изучены такие вопросы как скорость схождения декодера [20–22], критерии остановки декодера [23–25] и схемы обмена внешними вероятностями между компонентными

декодерами [26–28].

Хорошие характеристики турбокодов, простота кодера и его гибкость объясняют их популярность и широкое практическое использование. На данный момент турбокоды применяются в системах сотовой связи, таких как LTE, UMTS, CDMA2000, Wi-Max, в системах спутникового телевещания (DVB-RCS) и спутниковой связи. По этой причине исследования данных кодов не прекращаются.

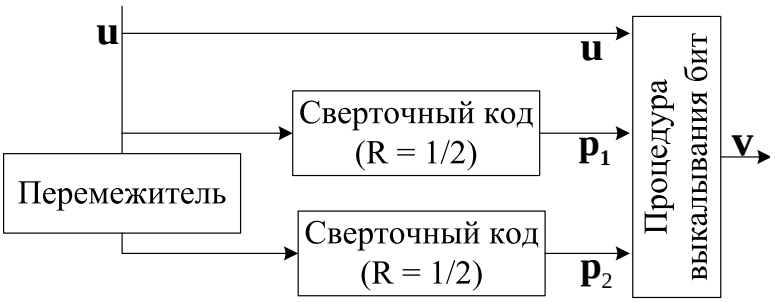

Рисунок 1.1 – Схема кодера турбокода

Прежде чем перейти к описанию турбокода, в разделе 1.2 будут описаны свойства и параметры сверточных кодов, которые являются неотъемлемой частью турбокода. Также рассмотрены алгоритмы декодирования сверточных кодов с жестким и мягким выходом, которые используются при декодировании турбокодов. Более детально структура турбокода и его свойства сформулированы в разделе 1.3. Раздел 1.4 посвящен алгоритмам декодирования турбокодов и их анализу. В разделе 1.5 приведены результаты моделирования различных декодеров в канале с АБГШ.

1.2 Сверточные коды

1.2.1 Структура сверточных кодов

Сверточные коды это полу-бесконечные линейные коды. В отличие от блоковых кодов, которые работают с блоками данных одинаковой длины, в общем виде сверточный код обрабатывает информационное слово произвольной длины,

а выходом кодера является полу-бесконечная последовательность закодированных значений. Обычно говорят, что скорость сверточного кода равна $R = b/c$, если на каждый блок из b входных бит, кодер генерирует c выходных значений. В этом случае входная и выходная последовательности соответственно обозначаются как $\mathbf{u} = \{\mathbf{u_0}, \mathbf{u_1}, \mathbf{u_2}, \ldots\}$ и $\mathbf{v} = \{\mathbf{v_0}, \mathbf{v_1}, \mathbf{v_2}, \ldots\}$, где $\mathbf{u_i} = \{u_i^{(0)}, \ldots, u_i^{(b-1)}\}$ и $\mathbf{v_i} = \{v_i^{(0)}, \ldots, v_i^{(c-1)}\}$.

Кодер сверточного кода может быть представлен как линейный регистр сдвига. Различают рекурсивные и нерекурсивные сверточные коды, которые могут быть представлены в виде линейного регистра сдвига с обратной связью и без обратной связи, соответственно. Выходные значения нерекурсивного кода зависят лишь от информационных бит, которые хранятся в ячейках памяти, в то время как в рекурсивном коде учитываются выходные биты.

В общем виде сверточный код со скоростью b/c выглядит как b регистров сдвига, связанных между собой сумматорами. Если обозначить как ν_i количество ячеек памяти в каждом из них, то суммарное значение $\nu = \sum_{i=1}^{b} \nu_i$ называется кодовым ограничением кода, а величина $\max_i\{\nu_i\}$ памятью кода. Таким образом, сложность кодера определяется количеством ячеек памяти в регистрах и количеством сумматоров. Пример регистра сдвига для систематического сверточного кода без обратной связи со скоростью $1/2$ приведен на рисунке 1.2.

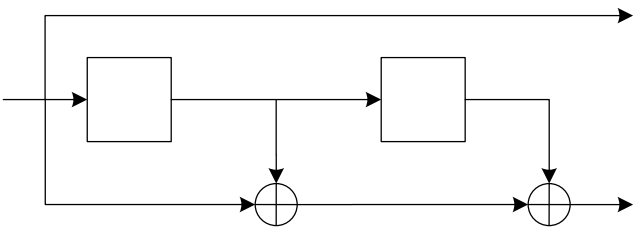

Рисунок 1.2 – Пример сверточного кода со скоростью $1/2$

Сверточный код также можно задать в полиномиальном виде. В этом случае входную и выходную последовательности записывают в виде многочленов,

которые выглядят как

$$u(x) = [u^{(0)}(x), u^{(2)}(x), \ldots, u^{(b-1)}(x)]$$

и

$$v(x) = [v^{(0)}(x), v^{(2)}(x), \ldots, v^{(c-1)}(x)],$$

где $u^{(j)}(x) = u_0^{(j)} + u_1^{(j)}x + u_2^{(j)}x^2 + u_3^{(j)}x^3 + \ldots$ и $v^{(j)}(x) = v_0^{(j)} + v_1^{(j)}x + v_2^{(j)}x^2 + v_3^{(j)}x^3 + \ldots$. Процесс кодирования можно представить как произведение полинома, который соответствует входной последовательности, на порождающую матрицу многочленов

$$u(x) = v(x)G(x). \tag{1.1}$$

Для получения порождающей матрицы многочленов, каждый регистр с соответствующими связями также записывается в виде многочлена. В общем виде он может быть представлен как

$$g(x) = \frac{f(x)}{q(x)} = \frac{f_0 + f_1x + f_2x^2 + \ldots + f_{\nu_i}x^{\nu_i}}{1 + q_1x + q_2x^2 + \ldots + q_{\nu_i}x^{\nu_i}},$$

где f_i коэффициенты прямой связи в регистре, а q_i коэффициенты обратной связи. Значение i-ого коэффициента равно 1, если в регистре есть отвод от i-ой ячейки памяти, в противоположном случае коэффициент равен 0. В случае отсутствия обратной связи в регистре $q(x) = 1$. Используя полиномиальную запись регистров, результирующая матрица будет выглядеть как

$$G(x) = \begin{pmatrix} g_{11}(x) & g_{12}(x) & \cdots & g_{1c}(x) \\ g_{21}(x) & g_{22}(x) & \cdots & g_{2c}(x) \\ \vdots & \vdots & \ddots & \vdots \\ g_{b1}(x) & g_{b2}(x) & \cdots & g_{bc}(x) \end{pmatrix}. \tag{1.2}$$

Для регистра, представленного на рисунке 1.2, полиномиальная порождающая матрица равна $G(x) = [1,\ 1 + x + x^2]$. Из полиномиальной формы матрицу можно переписать в двоичном виде. Тогда порождающая матрица кода будет

полу-бесконечной блочно-диагональной матрицей, которая для кода $[1,\ 1+x+x^2]$ может быть записана как

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix}. \tag{1.3}$$

Число возможных состояний регистра сдвига зависит от кодового ограничения кода и равно $N = 2^{\nu}$. Т.к. выходные значения кодера зависят лишь от входных бит и от текущего состояния регистра, то весь процесс кодирования можно представить как работу конечного автомата, который может быть представлен в виде диаграммы переходов. Из каждого состояния такого конечного автомата выходит 2^b ребер, на которых отложены значения соответствующих входных и выходных бит. Пример диаграммы для рассмотренного выше кода приведен на рисунке 1.3. Если развернуть диаграмму состояний во времени, то будет получена так называемая решетчатая диаграмма, пример которой приведен на рисунке 1.4. Каждой информационной последовательности соответствует путь или набор состояний на решетчатой диаграмме. Представление сверточного кода в виде решетчатой диаграммы очень удобно при декодировании. В этому случае задача декодера сводится к нахождению наиболее вероятного пути на решетке.

Как и для блоковых кодов, при анализе корректирующей способности сверточного кода рассматривают его минимальное расстояние. Но в отличие от блоковых кодов сверточный код характеризуется множеством минимальных расстояний. Т.к. длина последовательности на входе кодера может быть произвольной, то для кодовых последовательностей различной длины расстояние будет различным

$$d_k = \min\{d(\mathbf{v}, \mathbf{v}')\}, \tag{1.4}$$

где $\mathbf{v}$ и $\mathbf{v}'$ различные кодовые последовательности длины k, а $d()$ расстояние

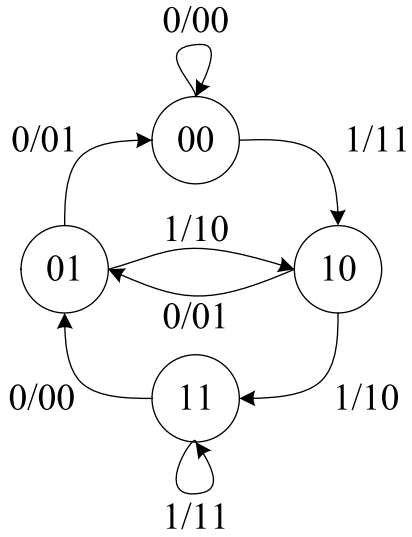

Рисунок 1.3 – Представление сверточного кода в виде диаграммы состояний

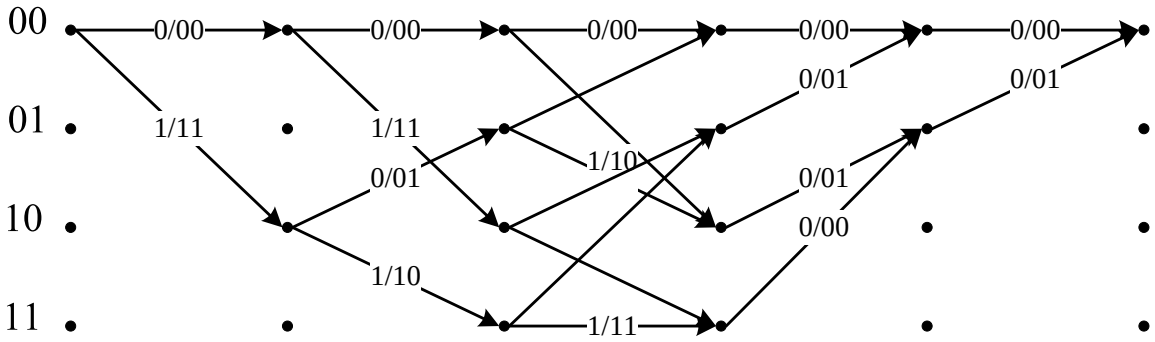

Рисунок 1.4 – Представление сверточного кода в виде решетчатой диаграммы

Хэмминга. Длина k называется шириной окна декодирования и чем больше его значение, тем больше ошибок может исправить код. Если рассматривать кодовые последовательности на решетчатой диаграмме, то очевидно, что с увеличением кодовой последовательности минимальное расстояние не уменьшается, т.е.

$$d_1 \leq d_2 \leq d_3 \leq \ldots \leq d_\infty. \tag{1.5}$$

Однако расстояние кода не растет бесконечно с увеличением k и в какой то момент достигает насыщения [29]. Наибольшее значение минимального расстояния называется свободным расстоянием кода и обозначается как

$$d_{free} = \max\{d_1, d_2, d_3, \ldots, d_\infty\}. \tag{1.6}$$

Другой важной характеристикой сверточного кода является спектр свободных расстояний кода, который определяет количество кодовых последовательностей, имеющих определенное значение свободного расстояния. В то время как для блоковых кодов существуют конструктивные методы нахождения хороших кодов (БЧХ, РС), для нахождения хороших сверточных кодов используют компьютерный поиск. Данный факт ограничивает возможность использования длинных сверточных кодов, т.к. их поиск является трудоемкой задачей. Списки хороших сверточных кодов можно найти в литературе [29].

Поскольку при использовании сверточных кодов в реальных системах связи входная последовательность делится на блоки определенной длины, встает задача корректного окончания кодирования. Самым простым подходом является использования усеченной кодовой конструкции. В данном случае путь на решетке сверточного кода начинается в нулевом состоянии, т.е. начальные значения памяти регистра равны нулю, и может заканчиваться в произвольном. Недостаток данного метода заключается в том, что при декодировании неизвестно конечное состояние пути на решетке, что приводит к менее надежному декодировании конечных бит информационного слова. Поэтому рассматривают две альтернативные конструкции, добавление нулевого хвоста (терминированная конструкция) и циклическое замыкание (кольцевая конструкция или тейлбайтинг). В терминированной конструкции к информационному слова добавляется ν бит, которые приводят путь на решетке в нулевое состояние. Но в этом случае приходится передавать дополнительные биты. Кольцевая конструкция лишена этого недостатка. При использовании циклического замыкания начальные значения в ячейках памяти сверточного кода инициализируются таким образом, чтобы начальные и конечные состояния пути на решетке совпадали. Например, для нерекурсивных кодов перед кодированием в ячейки регистра записываются ν последних бит информационной последовательности, тогда при завершении кодирования последних ν бит информационного слова начальные и конечные состояния регистра будут одинаковыми. Данной свойство позволяет надежно декодировать

последние биты информационного слова и при этом не досылать дополнительные значения.

С практической точки зрения наибольший интерес вызывают сверточные коды со скоростью $1/c$. Это вызвано тем, что механизм перфорации или выкалывания позволяет из сверточного кода со скоростью $1/c$ получить код с произвольной большей скоростью. К примеру, при удалении из кодового слова каждого c-ого бита мы получим код со скоростью $1/(c-1)$. При этом полученные таким образом высокоскоростные коды могут быть представлены с помощью одной и той же решетчатой диаграммы, а следовательно декодированы с помощью одного и того же алгоритма. Это позволяет иметь код с различными скоростями, который декодируется одним и тем же декодером.

1.2.2 Декодирование сверточных кодов

Основное внимание в данной работы уделено систематическим сверточным кодам со скоростью $1/c$ и при описании алгоритмов декодирования предполагается использование последних. Однако стоит отметить, что все выкладки справедливы и для не систематических кодов. Прежде чем перейти к их описанию введем следующие обозначения. Пусть $\mathbf{u} = [u_0, u_1, \ldots, u_{k-1}]$ информационная последовательность длины k, а $\mathbf{v} = [\mathbf{v_0}, \ldots, \mathbf{v_{k-1}}]$ выход кодера, где $\mathbf{v_i} = [v_i^{(0)}, v_i^{(1)}, \ldots, v_i^{(c-1)}]$ кодовые биты, соответствующие информационному биту u_i. Также обозначим как $\mathbf{a_i} = [a_i^{(0)}, a_i^{(1)}, \ldots, a_i^{(c-1)}]$, где $a_i^{(j)} = \sqrt{E_s}(2v_i^{(j)} - 1)$, сигнальное отображение бит кодового слова. Т.е. если $v_i^{(j)} = 1$, то $a_i^{(j)} = \sqrt{E_s}$, в противном случае $a_i^{(j)} = -\sqrt{E_s}$. Как E_s обозначена энергия затрачиваемая для передачи одного бита кодового слова, которая соотносится с энергий на бит информационного слова E_b как $E_s = RE_b$.

Схема рассматриваемой системы передачи данных представлена на рисунке 1.5. Обычно в качестве канала рассматривают канал с аддитивным белым Гауссовским шумом (АБГШ) [30]. Это связано с тем, что перемежитель коди-

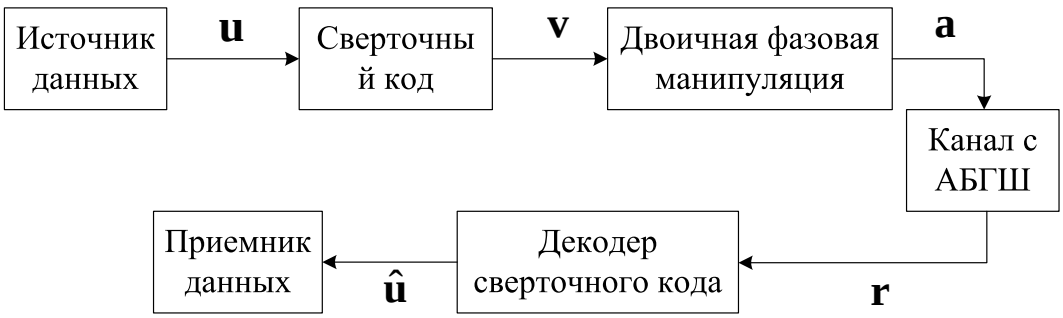

Рисунок 1.5 – Пример системы передачи данных

рованных бит, который учитывает характер ошибок в канале, переводит любые группы ошибок, вызванные помехами в канале, в независимые. При передаче через такой канал выходные значения равны $\mathbf{r_i} = [r_i^{(0)}, r_i^{(1)}, \ldots, r_i^{(c-1)}]$, где $r_i^j = a_i^{(j)} + n_i^{(j)}$, а $n_i^{(j)}$ независимая Гауссовская величина с нулевым математическим ожиданием и спектральной плотностью мощности шума равной $N_0/2$. Для такой величины плотность условной вероятности $P(r_i^{(j)}|v_i^{(j)})$ равна

$$P(r_i^{(j)}|v_i^{(j)}) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(r_i^{(j)} - a_i^{(j)})^2}, \tag{1.7}$$

где $\sigma^2 = N_0/2E_s$ дисперсия шума. Т.к. канал с АБГШ является каналом без памяти, то справедливо следующее выражение

$$P(\mathbf{r_i}|\mathbf{v_i}) = \frac{1}{(\sqrt{2\pi}\sigma)^c} e^{-\frac{1}{2\sigma^2}\|\mathbf{r_i} - \mathbf{a_i}\|^2}, \tag{1.8}$$

где $\|\cdot\|^2$ Евклидово расстояние.

При передаче через канала без памяти задача декодирования кода заключается в нахождении последовательности состояний в решетке сверточного кода, которая может быть представлена как Марковский процесс. Оптимальными декодерами сверточных кодов, которые основаны на их решетчатом представлении являются алгоритмы Витерби и BCJR. Декодер Витерби может работать как с жесткими, так и с мягкими входами, в то время как BCJR является декодером с мягким входом. Декодер Витерби является алгоритмом, который минимизирует вероятность ошибки на слово и выбирает декодированное слово $\mathbf{u}$ так, чтобы максимизировать вероятность $P(\mathbf{r}|\mathbf{u})$, т.е. является декодером по максимуму

правдоподобия (МП)

$$\hat{\mathbf{u}} = \arg\max_{\mathbf{u}} P(\mathbf{r}|\mathbf{u}),$$

где $\hat{\mathbf{u}}$ принятое решение о декодированном слове.

BCJR решает другую задачу и минимизирует вероятность ошибки на бит. Алгоритм является декодером по максимуму апостериорной вероятности и максимизирует вероятность $P(u_i|\mathbf{r})$

$$\hat{u}_i = \arg\max_{u_i} P(u_i|\mathbf{r}),$$

где $\hat{u}_i$ принятое решение о декодированном бите. В отличии от алгоритма Витерби BCJR вычисляет вероятности выходных бит декодированного информационного слова $P(\hat{u}_i|\mathbf{r})$. Но для алгоритма Витерби существует расширение, позволяющее рассчитывать приближенные вероятности декодированных бит, которое называется SOVA (Soft Output Viterbi Algorithm).

Мягкие решения декодера удобно представлять в виде логарифма отношения правдоподобий (LLR), которые также называют надежностями бит

$$L(\hat{u}_i) = \ln \frac{P(\hat{u}_i = 1|\mathbf{r})}{P(\hat{u}_i = 0|\mathbf{r})},$$

где $\ln()$ - натуральный логарифм. В случае, если $L(\hat{u}_i) > 0$, то $\hat{u}_i = 1$, в противном случае декодер принимает решение, что $\hat{u}_i = 0$. Для того, чтобы из значения надежности снова перейти в вероятностную область нужно воспользоваться свойством, что $P(\hat{u}_i = 1|\mathbf{r}) + P(\hat{u}_i = 0|\mathbf{r}) = 1$. Тогда значения $P(\hat{u}_i = 1|\mathbf{r})$ и $P(\hat{u}_i = 1|\mathbf{r})$ можно вычислить следующим образом

$$P(\hat{u}_i = 1|\mathbf{r}) = \frac{1}{1 + e^{-L(\hat{u}_i)}}, \tag{1.9}$$

$$P(\hat{u}_i = 0|\mathbf{r}) = \frac{e^{-L(\hat{u}_i)}}{1 + e^{-L(\hat{u}_i)}}. \tag{1.10}$$

Входом декодеров обычно также являются не вероятности, а надежности $L(r_i^{(j)}) = \ln \frac{P(r_i^{(j)}|v_i^{(j)}=1)}{P(r_i^{(j)}|v_i^{(j)}=0)}$. Для канала с АБГШ

$$L(r_i^{(j)}) = L_c r_i^{(j)}, \tag{1.11}$$

где $L_c = \frac{2\sqrt{E_s}}{\sigma^2}$. Для перехода в вероятностную область также можно воспользоваться формулами 1.9 и 1.10.

Алгоритм Витерби

Декодер Витерби был предложен в 1967 году А. Витерби [31], однако его текущая интерпретация и доказательство оптимальности были даны Д. Д. Форни Мл. в [32]. Как уже было сказано выше, алгоритм Витерби минимизирует вероятность ошибки на слово, что эквивалентно максимизации вероятности $P(\mathbf{u}|\mathbf{r})$. Используя правило Байеса, эта вероятность может быть выражена как

$$P(\mathbf{u}|\mathbf{r}) = \frac{P(\mathbf{r}|\mathbf{u})P(\mathbf{u})}{P(\mathbf{r})}.$$

Значение $P(\mathbf{r})$ одинаково для всех возможных $\mathbf{u}$ и его можно опустить, т.к. оно является нормирующим коэффициентом. Также во многих случаях значения u_i считают равновероятными, что приводит к тому, что задача максимизации $P(\mathbf{u}|\mathbf{r})$ сводится к задаче максимизации $P(\mathbf{r}|\mathbf{u})$, которую решает декодер Витерби. Однако в случае, когда u_i не равновероятны, то $P(\mathbf{u})$ необходимо учитывать

Т.к. $\mathbf{u}$ и $\mathbf{v}$ однозначно задают путь на решетке, то $P(\mathbf{r}|\mathbf{u}) = P(\mathbf{r}|\mathbf{v})$ и для канала без памяти может быть вычислена как

$$P(\mathbf{r}|\mathbf{v}) = \Pi_{i=0}^{k-1} P(\mathbf{r_i}|\mathbf{v_i}). \tag{1.12}$$

Для канала с АБГШ $P(\mathbf{r_i}|\mathbf{v_i})$ задается формулой 1.8. С практической точки зрения удобнее работать в логарифмической области. В этом случае выражение 1.12 примет вид

$$\ln P(\mathbf{r}|\mathbf{v}) = \sum_{i=0}^{k-1} \ln(P(\mathbf{r_i}|\mathbf{v_i})). \tag{1.13}$$

Применительно к сверточным кодам $-\ln P(\mathbf{r}|\mathbf{v})$ называется метрикой пути и обозначается как $M(\mathbf{r}|\mathbf{v})$, а $-\ln P(\mathbf{r_i}|\mathbf{v_i})$ называется метрикой ребра и обозначается как $M(\mathbf{r_i}|\mathbf{v_i})$. Для канала с АБГШ метрика ребра выводится из выражения 1.8 и имеет следующий вид

$$M(\mathbf{r_i}|\mathbf{v_i}) = -c\ln(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \parallel \mathbf{r_i} - \mathbf{a_i} \parallel^2 . \tag{1.14}$$

Рассмотрим секцию решетки с номером i. В случае двоичного сверточного кода со скоростью $1/c$ в каждое состояние решетки входит два ребра, соответствующие входным информационным битам 0 и 1. Назовем частичной метрику такого пути, который начинается в нулевой секции решетки и заканчивается в некотором состоянии секции i. Обозначим такую частичную метрику для двух рассматриваемых путей, которые в секции $i-1$ заканчиваются в состоянии p_1 и p_2 (рисунок 1.6) соответственно как $M_{i-1}(p_1)$ и $M_{i-1}(p_2)$. Тогда метрика путей, которые из состояний p_1 и p_2 переходят в состояние q, может быть вычислена как

$$M_i(q) = M_{i-1}(p_1) + M(\mathbf{r_i}, \hat{\mathbf{v}}_\mathbf{i}^{(\mathbf{p_1},\mathbf{q})}) \tag{1.15}$$

или

$$M_i(q) = M_{i-1}(p_2) + M(\mathbf{r_i}, \hat{\mathbf{v}}_\mathbf{i}^{(\mathbf{p_2},\mathbf{q})}), \tag{1.16}$$

где $M(\mathbf{r_i}, \hat{\mathbf{v}}_\mathbf{i}^{(\mathbf{p},\mathbf{q})})$ метрика перехода (p, q), которому соответствуют некоторые выходные кодовые биты, обозначенные как $\hat{\mathbf{v}}_\mathbf{i}^{(\mathbf{p},\mathbf{q})}$.

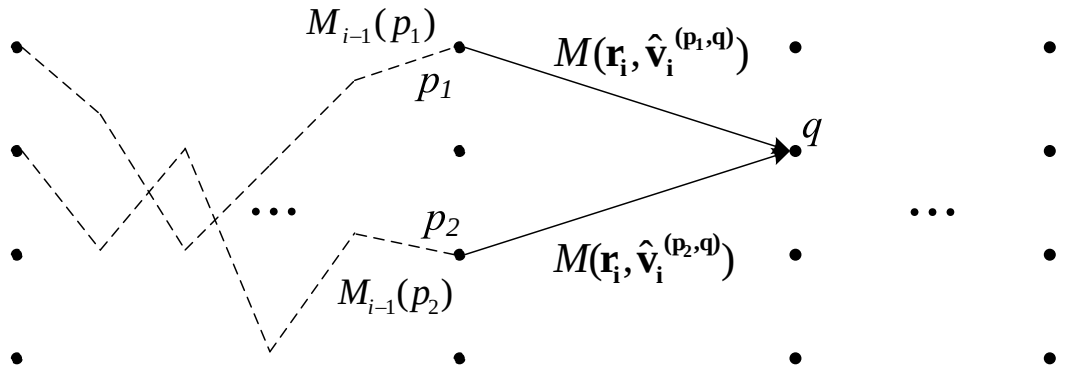

Рисунок 1.6 – Пример работы декодера Витерби на решетке сверточного кода

Основной целью алгоритма Витерби является максимизация суммарной метрики пути. Поэтому данный алгоритм руководствуется принципом оптимальности Беллмана, который для данного случая можно сформулировать следующим образом [33]: «Для нахождения кратчайшего пути на решетке, путь входящий в состояние q должен быть самым коротким. Иначе можно будет найти более короткий путь на решетке». Таким образом, основным правилом алгоритма Витерби является выбор пути с максимальным значением частичной метрики

в каждом состоянии решетки. Такой путь называется выжившим, а его метрика будет вычислена как

$$M_i(q) = \max\{M_{i-1}(p_1) + M(\mathbf{r_i}, \hat{\mathbf{v}}_\mathbf{i}^{(\mathbf{p_1},\mathbf{q})}), M_{i-1}(p_2) + M(\mathbf{r_i}, \hat{\mathbf{v}}_\mathbf{i}^{(\mathbf{p_2},\mathbf{q})})\}. \quad (1.17)$$

Т.к. до конца решетки не известно, какой из путей выживет и будет лучшим, то в каждом состоянии решетки необходимо хранить частичную метрику выжившего пути, и номер состояния, из которого пришел этот путь. Когда данные действия будут проделаны для последней секции решетки, то в результате будет найден лучший путь. Для того, чтобы восстановить информационное слово, необходимо выполнить обратный проход по решетке. Поскольку в каждом состоянии сохранен номер состояния из которого пришел лучший путь, то это может быть легко проделано. Также стоит учесть, что сложность алгоритма имеет экспоненциальную зависимость от значения кодового ограничения, т.к. вычисление метрик путей и выбор лучшего пути происходит в каждом состоянии решетки.

В общем виде алгоритм Витерби описывается следующим образом:

1. Если известно начальное состояние пути на решетке сверточного кода, то установить $i = 0$, $M_0(0) = 0$ и $M_0(j) = -\infty$, для всех $j = 1, \ldots, N-1$. Если начальное состояние не известно,то все $M_0(j) = 0$, для всех $j \in \{0, \ldots, N-1\}$.
2. Увеличить i на единицу и рассчитать метрики переходов.
3. Для каждого состояния q найти метрики входящих путей по формулам 1.15 и 1.16.
4. Выбрать путь с максимальной метрикой для состояния q в соответствии с выражением 1.17.
5. Сохранит состояние p из которого пришел путь с максимальной частичной метрикой.
6. Повторить шаги 2-6 для всех состояний решетки.

7. Если известно, что решетка замыкается в состоянии 0:
   (a) Из конечного состояния выполнить обратный проход в решетке по лучшему пути и найти соответствующее информационное слово
8. Если конечное состояние пути на решетке не известно:
   (a) Найти конечное состояния с максимальной метрикой пути и из него выполнить обратный проход по решетке для восстановления информационного слова.

При работе алгоритма Витерби в канале с АБГШ в логарифмической области удобнее пользоваться другой метрикой пути и ребра, а именно корреляцией между принятой последовательностью и декодируемыми битами. Тогда метрика перехода i-ой секции, которому соответствуют некоторые кодовые биты $[\hat{v}_i^0, \hat{v}_i^{(1)}, \ldots, \hat{v}_i^{(c-1)}]$, рассчитывается как

$$M_{corr}(\mathbf{r_i}, \mathbf{v_i}) = \sum_{j=1}^{c} r_i^{(j)} \hat{v}_i^{(j)}. \tag{1.18}$$

Частичные метрики путей рассчитываются также, как было показано выше.

Алгоритм SOVA

В 1989 году Хагенауер и Хоехер [12] предложили алгоритм Витерби с мягким выходом (SOVA), который приближенно рассчитывает надежности декодированных бит. Основные шаги работы алгоритма повторяют алгоритм Витерби с небольшими отличиями. Как и в алгоритме Витерби SOVA выполняет прямой и обратный проходы по решетке для нахождения декодированного бита. Однако, помимо этого, выполняются еще k частичных обратных проходов, в течении которых вычисляются надежности бит. В каждом состоянии алгоритм SOVA помимо информации о лучшем пути сохраняет разницу частичных метрик между выжившим путем и тем, который был отброшен

$$\begin{aligned} \Delta_i^q &= \max(M_{i-1}(p_1) + M(\mathbf{r_i}, \hat{\mathbf{v}}_\mathbf{i}^{(\mathbf{p_1},\mathbf{q})}), M_{i-1}(p_2) + M(\mathbf{r_i}, \hat{\mathbf{v}}_\mathbf{i}^{(\mathbf{p_2},\mathbf{q})})) \\ &- \min(M_{i-1}(p_1) + M(\mathbf{r_i}, \hat{\mathbf{v}}_\mathbf{i}^{(\mathbf{p_1},\mathbf{q})}), M_{i-1}(p_2) + M(\mathbf{r_i}, \hat{\mathbf{v}}_\mathbf{i}^{(\mathbf{p_2},\mathbf{q})})). \end{aligned} \tag{1.19}$$

В [12] показано, что разности Δ_i^q соответствует надежность корректности выбора выжившего пути. А т.к. выживший и отброшенный пути на некотором участке имеют различные соответствующие информационные битовые последовательности, то эта метрика может быть использована для вычисления надежностей последних.

Пусть начальные и конечные состояния на решетке равны 0, тогда все пути на решетке имеют общее начальное и конечное состояния. Рассмотрим пути, которые использовались для вычисления Δ_i^q и обозначим как $l = i - \delta$ секцию решетки в которой они разошлись. Обычно значение δ не превышает 5-6 кодовых ограничений кода. На участке от l-ой до i-ой секций рассматриваемые пути имеют различные переходы и, соответственно, различные информационные битовые последовательности. Если в секции i выбор лучшего пути был сделан неверно, то в позициях, где биты двух путей различаются, произойдет ошибка, вероятность которой будет зависеть от значения Δ_i^q. Т.к. в каждом состоянии лучшего пути алгоритм SOVA отбрасывает путь с меньшей метрикой и надежность каждого бита зависит от ряда отброшенных путей, то итоговые значения надежностей бит, которые различны у лучшего пути и отброшенного в состоянии q, будут вычислены как

$$L(\hat{u}_j) = \min\{L(\hat{u}_j), \Delta_i^q\} \tag{1.20}$$

для всех $j \in \{l, \ldots, l + \delta\}$, где биты различны и где начальное значение надежности бита равно $L(\hat{u}_j) = \infty$. Знак надежности задается значением бита $\hat{u}_i$, т.е. знак отрицательный если $\hat{u}_i = 0$, и положительный если $\hat{u}_i = 1$. Таким образом, алгоритм SOVA описывается следующими шагами:

1. Выполнить алгоритма Витерби для нахождения пути с максимальной метрикой и сохранить разность Δ_i^q для каждого состоянии лучшего пути по формуле 1.19.
2. Установить надежность каждого бита равной $L(\hat{u}_i) = \infty$.
3. Для вычисления надежностей выходных бит:

(a) Из каждого состояния лучшего пути выполнить обратный проход вдоль пути, который был отброшен.

(b) Для бит, которые различаются у лучшего пути и у отброшенного, вычислить надежность бита по формуле 1.20.

4. Знак надежности выставить в соответствии со значением декодированного бита.

Для коррекции значений надежностей на выходе алгоритма SOVA в работе [34] был предложен дальнейший способ улучшения. В основе алгоритма лежит тот факт, что если запустить алгоритм SOVA с начала решетки и с конца, то декодированные слова будут одинаковыми, однако рассчитанные значения надежностей будут разными. Таким образом, можно получить две последовательности надежностей бит, начиная декодирование с начала и конца решетки. В качестве результирующей надежности бита выбирается минимальное значение из двух полученных последовательностей. Несмотря на то, что подход позволяет более точно рассчитать выходные надежности бит, он редко применяется на практике и не будет рассматриваться в дальнейшем.

Алгоритм BCJR

Алгоритм BCJR был предложен Л. Балем и др. в работе [11] как алгоритм работающий по максимуму апостериорной вероятности для нахождения вероятностей состояния и/или бита в Марковских цепях с конечных числом состояний. Т.к. выход сверточного кода, прошедший через канал без памяти, можно считать Марковским источником, то возможно использование данного алгоритма для его декодирования. Алгоритм BCJR вычисляет апостериорные вероятности каждого перехода в решетке, которые в дальнейшем используются для вычисления апостериорной вероятности бит

Разделим выход канала на три части: $\mathbf{r}_{<\mathbf{i}}$ - символы, соответствующие секциям до момента времени i; $\mathbf{r}_{\mathbf{i}}$ - символы, соответствующие секции решетки с номером i; $\mathbf{r}_{>\mathbf{i}}$ - символы, соответствующие «будущим» секциям решетки. Рассмотрим секцию решетки сверточного кода в момент времени i (рисунок 1.7),

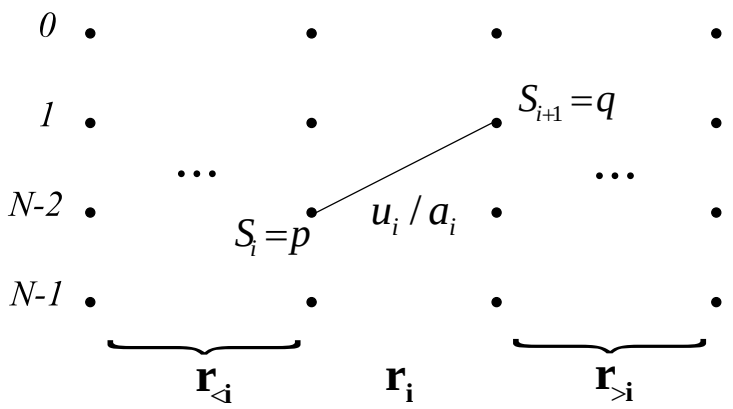

Рисунок 1.7 – Переход (p, q) в решетке сверточного кода

где изображен переход из состояния $S_i = p$ в состояние $S_{i+1} = q$. Апостериорная вероятность перехода (p, q) с учетом того, что $\mathbf{r} = (\mathbf{r_{<i}}, \mathbf{r_i}, \mathbf{r_{>i}})$ может быть выражена как

$$P(S_i = p, S_{i+1} = q|\mathbf{r}) = \frac{p(S_i = p, S_{i+1} = q, \mathbf{r_{<i}}, \mathbf{r_i}, \mathbf{r_{>i}})}{p(\mathbf{r})}. \tag{1.21}$$

Данное выражение можно переписать в развернутом виде, используя то факт, что источник Марковский

$$P(S_i = p, S_{i+1} = q|\mathbf{r}) = \frac{p(S_i = p, \mathbf{r_{<i}})p(S_{i+1} = q, \mathbf{r_i}|S_i = p)p(\mathbf{r_{>i}}|S_{i+1} = q)}{p(\mathbf{r})}. \tag{1.22}$$

Первое значение в числителе показывает вероятность того, что путь на решетке закончится в состоянии p в момент времени i с учетом принятой последовательности $\mathbf{r_{<i}}$. Второе значение это вероятность того, что путь на решетке проходит через состояния (p, q) с учетом принятого значения $\mathbf{r_i}$. Третье выражение это вероятность того, что путь на решетке начинается в состоянии q в момент времени i с учетом принятых из канала символов $\mathbf{r_{>i}}$. Обычно эти вероятности обозначают как

$$\alpha_i(p) = p(S_i = p, \mathbf{r_{<i}}), \tag{1.23}$$

$$\gamma_i(p, q) = p(S_{i+1} = q, \mathbf{r_i}|S_i = p), \tag{1.24}$$

$$\beta_{i+1}(q) = p(\mathbf{r_{>i}}|S_{i+1} = q). \tag{1.25}$$

Тогда выражение для нахождения апостериорной вероятности может быть представлено в более лаконичном виде

$$P(S_i = p, S_{i+1} = q|\mathbf{r}) = \frac{\alpha_i(p)\gamma_i(p,q)\beta_{i+1}(q)}{p(\mathbf{r})}. \tag{1.26}$$

Прежде чем перейти к способу расчета значений 1.23, 1.24, 1.25 предположим, что для i-ой секции решетки рассчитаны все апостериорные вероятности для всех переходов. Разделим все переходы в i-ой секции на два множества. Те, которые были инициированы единичным информационным битом, обозначим как Q_1, а те, которые соответствуют нулевому информационному биту, обозначим как Q_0. Тогда апостериорная вероятность бита будет вычислена следующим образом

$$\begin{aligned} P(\hat{u}_i = x|\mathbf{r}) &= \sum_{(p,q)\in Q_x} P(S_i = p, S_{i+1} = q|\mathbf{r}) = \\ &= \frac{1}{p(\mathbf{r})} \sum_{(p,q)\in Q_x} \alpha_i(p)\gamma_i(p,q)\beta_{i+1}(q), \end{aligned} \tag{1.27}$$

где $x \in \{0, 1\}$.

Для вычисления значений $\alpha_{i+1}(q)$ и $\beta_i(p)$ используются рекуррентные выражения. Предположим, что известны значения $\alpha_i(p)$ для всех состояний $p \in \{0, \ldots, N-1\}$ в i-ой секции решетки, тогда $\alpha_{i+1}(q)$ может быть вычислена как

$$\alpha_{i+1}(q) = \sum_{p=0}^{N-1} \alpha_i(p)\gamma_i(p,q). \tag{1.28}$$

Значения $\beta_i(p)$ рассчитываются по тому же принципу. Отличие заключается в том, что вычисления для $\alpha_{i+1}(q)$ начинаются с начала решетки, в то время как для $\beta_i(p)$ вычисление производится с ее конца. Предположим, что известны значения $\beta_{i+1}(q)$ для всех состояний $q \in \{0, \ldots, N-1\}$, тогда $\beta_i(p)$ может быть вычислена как

$$\beta_i(p) = \sum_{q=0}^{N-1} \beta_{i+1}(q)\gamma_i(p,q) \tag{1.29}$$

Процесс вычисления значений $\alpha_{i+1}(q)$ и $\beta_i(p)$ проиллюстрирован на рисунке 1.8.

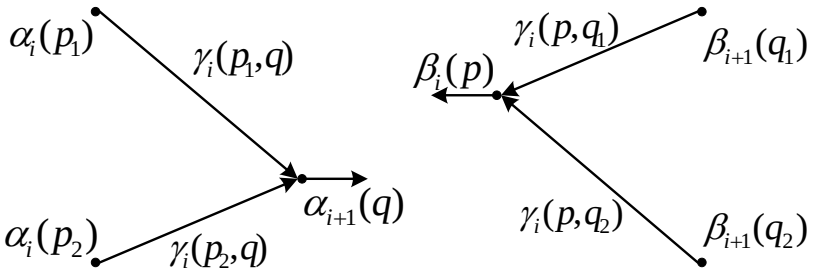

Рисунок 1.8 – Иллюстрация вычисления значений $\alpha_{i+1}(q)$ и $\beta_i(p)$

Начальные значения $\alpha_0(0)$ и $\beta_k(0)$ зависят от сверточного кода. Если известно, что начальное и конечное состояния пути на решетке сверточного кода равны 0, то $\alpha_0(0) = 1$ и $\beta_k(0) = 1$, в то время как для остальных начальных состояний они равны 0. Если конечное и начальное состояния кодера не известны, то при декодировании все конечные и начальные состояния считаются равновероятными и равными $1/N$.

Расчет значений α и β для всех состояний решетки выполняется в ходе прямого и обратного прохода по решетке сверточного кода. Как видно из выражений 1.28 и 1.29 неизвестной величиной остается вероятность перехода (p, q), т.е. $\gamma_i(p, q)$. Способ расчета значений $\gamma_i(p, q)$ зависит от значений принятых из канала и от типа канала. Учтем то, что переход (p, q) в i-ой секции решетки однозначно задается входным битом $u^{(p,q)}$ и однозначно задает выходные значения $\mathbf{v}^{(\mathbf{p},\mathbf{q})}$. Тогда для кода со скоростью $R = 1/c$ и канала с АБГШ значение $\gamma_i(p, q)$ может быть вычислено как

$$\gamma_i(p, q) = \frac{1}{(\sqrt{2\pi}\sigma)^c} e^{-\frac{1}{2\sigma^2}\|\mathbf{r_i} - \hat{\mathbf{a}}_\mathbf{i}^{(\mathbf{p},\mathbf{q})}\|^2} P(u_i = x), \tag{1.30}$$

где $P(u_i = x)$ априорная вероятность того, что u_i равно $x \in \{0, 1\}$. Обычно она равна $1/2$, однако в случае турбо-декодирования это не так и именно эта вероятность передается от одного декодера к другому.

Таким образом, алгоритм BCJR сводится к следующим шагам:

1. Если известно, что начальное и конечное состояние в решетке равно 0, то установить $\alpha_0(0) = 1$ и $\beta_k(0) = 1$, а $\alpha_0(q) = 0$ и $\beta_k(q) = 0$ для всех

$q \in \{1, \ldots, N-1\}$. В противном случае $\alpha_0(q) = 1/N$ и $\beta_k(q) = 1/N$ для всех $q \in \{0, \ldots, N-1\}$.

2. Вычислить метрики переходов $\gamma_i(p, q)$ для всех ребер по формуле 1.30.
3. Вычислить метрики состояний $\alpha_i(p)$ в ходе прямого прохода по решетке по формуле 1.28.
4. Вычислить метрики состояний $\beta_{i+1}(q)$ в ходе обратного прохода по решетке по формуле 1.29.
5. Вычислить апостериорные вероятности выходных бит по формуле 1.27.

Алгоритм Log-MAP

Недостатком алгоритма BCJR является то, что необходимо выполнять большое число умножений. Переход к логарифмическому представлению позволяет заменить операцию умножения сложением и уменьшить сложность алгоритма при той же точности и производительности. В логарифмическом виде выходные значения алгоритма Log-MAP примут следующий вид

$$L(\hat{u}_i) = \ln \frac{P(\hat{u}_i = 1|\mathbf{r})}{P(\hat{u}_i = 0|\mathbf{r})} = \ln \frac{\sum\limits_{(p,q)\in Q_1} \alpha_i(p)\gamma_i(p,q)\beta_{i+1}(q)}{\sum\limits_{(p,q)\in Q_0} \alpha_i(p)\gamma_i(p,q)\beta_{i+1}(q)}. \tag{1.31}$$

Если определить $\ln(\alpha_i(p))$, $\ln(\beta_{i+1}(q))$, $\ln(\gamma_i(p,q))$ как $A_i(p)$, $B_{i+1}(q)$ и $\Gamma_i(p,q)$, соответственно, то выражение 1.31 можно записать как

$$\begin{aligned} L(\hat{u}_i) &= \ln \left(\sum_{(p,q)\in Q_1} e^{A_i(p)+\Gamma_i(p,q)+B_{i+1}(q)} \right) \\ &- \ln \left(\sum_{(p,q)\in Q_0} e^{A_i(p)+\Gamma_i(p,q)+B_{i+1}(q)} \right). \end{aligned} \tag{1.32}$$

Вычисление метрик $A_{i+1}(q)$ и $B_i(p)$ будет выглядеть следующим образом

$$A_{i+1}(q) = \ln \left(\sum_{p=0}^{Q-1} e^{A_i(p)+\Gamma_i(p,q)} \right), \tag{1.33}$$

$$B_i(p) = \ln \left(\sum_{q=0}^{Q-1} e^{B_{i+1}(q)+\Gamma_i(p,q)} \right). \tag{1.34}$$

Для того, чтобы рассчитать метрику перехода в логарифмической области $\Gamma_i(p,q)$ воспользуемся выражением 1.30, тогда

$$\Gamma_i(p,q) = -c\ln(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \parallel \mathbf{r_i} - \hat{\mathbf{a}}_\mathbf{i}^{(\mathbf{p},\mathbf{q})} \parallel^2 + \ln P(u_i = x). \tag{1.35}$$

При расчете метрики ребра постоянной $-c\ln(\sqrt{2\pi}\sigma)$ можно пренебречь, т.к. она сокращается при вычислении окончательной надежности бита.

Алгоритм Max-Log-MAP

Тем не менее Log-MAP является вычислительно сложным и требует вычисления экспонент и логарифмов. Для уменьшения сложности декодера на практике часто пользуются приближенным способом расчета метрик состояний и путей, за счет приблизительного вычисления логарифма суммы экспонент:

$$\ln(\sum_i e^{x_i}) \approx \max_i x_i. \tag{1.36}$$

Алгоритм, использующий это правило для расчета метрик, называется Max-Log-MAP. Расчет метрик $A_{i+1}(q)$ и $B_i(p)$ сводится к следующим несложным правилам

$$A_{i+1}(q) \approx \max_p(A_i(p) + \Gamma_i(p,q)), \tag{1.37}$$

$$B_i(p) \approx \max_q(B_{i+1}(q) + \Gamma_i(p,q)). \tag{1.38}$$

При вычислении апостериорных вероятностей бит также можно воспользоваться приближением 1.36, тогда

$$\begin{aligned} L(\hat{u}_i) &= \max_{(p,q)\in Q_1}(A_i(p) + \Gamma_i(p,q) + B_{i+1}(q)) \\ &- \max_{(p,q)\in Q_0}(A_i(p) + \Gamma_i(p,q) + B_{i+1}(q)). \end{aligned} \tag{1.39}$$

Плюсом алгоритма Max-Log-MAP является его небольшая вычислительная сложность. Также стоит отметить, что для работы алгоритма Log-MAP необходимо учитывать значение дисперсии канала с АБГШ. Для алгоритма Max-Log-MAP это не важно, т.к. для вычисления значений всех метрик используется только операция суммирования и коэффициентом L_c можно пренебречь. Однако

вычисленные апостериорные вероятности бит являются лишь приближенными значениями, что негативно сказывается, например, при декодировании турбокода. Для уточнения значения вычисленных надежностей в выражении 1.36 можно использовать корректирующий коэффициент, тогда выражение 1.36 примет вид

$$\ln(e^{x_1} + e^{x_2}) = \max(x_1, x_2) + f(x_1, x_2). \tag{1.40}$$

Впервые данный метод был предложен П. Робертсоном и др. в [13]. Функция $f(x_1, x_2)$ является коррекцией и если она равна $\ln(1 + e^{-|x_1+x_2|})$, то это эквивалентно использованию алгоритма Log-MAP, и позволяет вместо вычисления двух экспонент вычислять лишь одну. Однако для того, чтобы не рассчитывать значения экспонент в принципе, авторы предлагают хранить в таблице небольшое число корректирующих коэффициентов, которые выбираются в зависимости от конкретных значений x_1 и x_2.

1.3 Структура кодера турбокода

Как видно на рисунке 1.1, кодер турбокода состоит из двух рекурсивных систематических сверточных (Recursive Systematic Convolutioncal, RSC) кодов со скоростью $R = 1/2$, связанных между собой k-битовым перемежителем. Сверточные коды, которые используются в процессе кодирования турбокода, часто называют компонентными кодами. Выбор именно рекурсивных кодов связан с тем, что вес кодового слова RSC кода сильно зависит от расположения единиц в информационном слове, в отличии от нерекурсивного, и он обеспечивает больший вес выходного слова при малом весе входного. В свою очередь турбокоды, использующие RSC коды, имеют большее минимальное расстояние и лучшие корректирующие способности.

В оригинальной статье [10] турбокод имеет скорость $R = 1/3$, а его кодовое слово выглядит как $\mathbf{v} = (\mathbf{u}, \mathbf{p_1}, \mathbf{p_2})$, где $\mathbf{p_1}$ и $\mathbf{p_2}$ проверочные биты первого и второго сверточных кодов, соответственно. В общем виде возможно исполь-

зование большего числа компонентных кодов, но принцип построения кода не изменится. Компонентные коды могут быть как различными, так и одинаковыми, однако, опыт показывается, что использование различных кодов не дает выигрыша.

Прежде чем поступить на вход второго сверточного кода, информационные биты проходят через перемежитель. Допустим, что вес Хэмминга последовательности $\mathbf{u}$ равен t, тогда полученная на выходе перемежителя последовательность $\Pi(\mathbf{u})$ также будет иметь вес t. Функция $\Pi()$ задает перемежитель. Однако проверочные биты на выходе каждого компонентного кода будут различными и будут иметь различный вес. Следовательно, если одна из проверочных последовательностей имеет малый вес, можно рассчитывать на то, что вторая проверочная последовательность будет большего веса и общее расстояние кода будет больше. Также стоит отметить, что коррелированные последовательности на входах компонентных декодеров негативно влияют на производительность турбо-декодера в целом. Перемежитель декоррелирует входные последовательности и позволяет использовать итеративный декодер. Таким образом, перемежитель решает две задачи: позволяет создавать коды с хорошим расстоянием и дает возможность использовать эффективный итеративный декодер.

Длина перемежителя в значительной степени влияет на производительность кода и чем она больше, тем лучше код [35]. Существует несколько видов перемежителей. Наиболее простым с практической точки зрения является блоковый перемежитель [36], который представляется в виде матрицы. Для перестановки входные биты построчно записываются в матрицу по строкам и вычитываются по столбцам. Также значения могут быть вычитаны по диагонали. Другой вид перемежителей называется случайным. Чтобы задать его, генерируются n чисел в случайном порядке и в соответствии с получившейся последовательностью происходит перестановка бит. По такому же принципу работает S-случайный перемежитель, для которого последовательность целых чисел генерируется по специальным правилам [37]. В [17] сказано, что для коротких слов

лучше использовать неслучайные перемежители, в то время как для больших длин случайные предпочтительнее.

Как и в случае сверточных кодов, для турбокодов возможно использовать механизм выкалывания бит для получения кода с большей скоростью. После выкалывания скорость кода вычисляется как $R = k/(k + y)$, где y количество проверочных бит, оставшихся после выкалывания. При этом структура декодера не изменяется, что является важным качеством и благодаря чему коды имеют широкое практическое применение.

Как было сказано выше, турбокоды имеют плохое минимальное расстояние d_{min}, однако имеют хорошие корректирующие способности. Это связано с тем, что кодовых слов веса d_{min} в турбокоде мало и основная их доля приходится на слова большего веса. Поэтому в турбокодах большее значение имеет среднее расстояние кода и распределение весов кодовых слов. По этой причине общей чертой, характеризующей все турбокоды, является резкий спад графика зависимости вероятности ошибки на бит (P_b) от отношения сигнал шум, после чего скорость спада резко уменьшается. Данная область называется областью насыщения вероятности ошибки или «полкой». Одним из способов уменьшения высоты полки является выбор хорошего перемежителя [35].

Т.к. длина перемежителя конечна и определяет длину входного информационного слова, то можно определить вес каждого кодового слова d, количество слова такого веса N_d и суммарный вес Хэмминга соответствующих им информационных слов w_d. В [38] получена аддитивная граница вероятности ошибки P_b турбокода:

$$P_b \leq \sum_{d=d_{free}}^{\infty} \frac{w_d}{k} Q\left(\sqrt{\frac{2RdE_b}{N_0}}\right), \tag{1.41}$$

где d_{free} свободное расстояние кода и k информационная длина. Часто на практике необходимо оценить высоту области насыщения вероятности ошибки. Это может быть сделано с помощью моделирования, однако если данная область оценивается, к примеру, для $P_b \leq 10^{-9}$, то моделирование становится слиш-

ком долгим. Зная эти значения и учитывая то, что полка обусловлена кодовыми словами малого веса, возможно оценить высоту полки с помощью следующего выражения [39]

$$P_b \approx \frac{w_{free}}{k} Q\left(\sqrt{\frac{2Rd_{free}E_b}{N_0}}\right), \tag{1.42}$$

где w_{free} суммарный вес информационных слов, которым соответствуют кодовые слова веса d_{free}. Для получения оценки работы итеративного турбо-декодера обычно к вычисленному таким образом значению добавляется 0,5 дБ для коррекции, что связано с подоптимальностью итеративного декодера [39]. Но получающиеся аналитические оценки достаточно плохо предсказывают поведение кода в области спада вероятности ошибки, и они не могут быть использованы для сравнения турбокодов между собой. Для этого используется моделирование.

1.4 Декодирование турбокода

Рассмотрим итеративную схему декодирования, которая была предложена в [10]. Как было сказано выше, ядром схемы являются два декодера сверточных кодов с мягким выходом. Представим принятое из канала слово как $\mathbf{r} = (\mathbf{r_s}, \mathbf{r_{p_1}}, \mathbf{r_{p_2}})$, где $\mathbf{r_s}$ соответствует принятой систематической части кодового слова, а $\mathbf{r_{p_1}}$ и $\mathbf{r_{p_2}}$ его проверочные символы. Тогда $(\mathbf{r_s}, \mathbf{r_{p_1}})$ и $(\Pi(\mathbf{r_s}), \mathbf{r_{p_2}})$ являются входом первого и второго компонентного декодера, соответственно. В своей работе К. Берроу и др. [10] показали, что для систематического сверточного кода выходные значения $L(\hat{u}_i)$ алгоритма Log-MAP, могут быть представлены как:

$$L(\hat{u}_i) = L_{sys}(\hat{u}_i) + L_{app}(\hat{u}_i) + L_{ext}(\hat{u}_i), \tag{1.43}$$

где $L_{sys}(\hat{u}_i)$, $L_{app}(\hat{u}_i)$, $L_{ext}(\hat{u}_i)$ соответственно систематическая, априорная и внешняя составляющие надежности. Вычисление априорных надежностей для следующего декодера происходит по правилу, которое следует из формулы 1.43:

$$L_{ext}(\hat{u}_i) = L(\hat{u}_i) - L_{sys}(\hat{u}_i) - L_{app}(\hat{u}_i). \tag{1.44}$$

Априорная или внутренняя составляющая это информация, которая известна о битах еще до декодирования. В случае итеративного декодирования турбокода априорные вероятности один декодер получает от второго. Систематическая или апостериорная составляющая надежности бита это информация, полученная из канала, т.е. $L_{sys}(\hat{u}_i) = L(r_s^{(i)})$, где $r_s^{(i)}$ это i-ый символ, соответствующий систематической части принятого кодового слова. Внешнюю информацию $L_{ext}(\hat{u}_i)$ декодер вычисляет исходя из принятых из канала значений и априорной информации, однако без учета $r_s^{(i)}$ и соответствующей i-ому символу априорной информации. Именно эта составляющая является априорной для следующего декодера.

Детальная схема итеративного декодера турбокода, который использует алгоритм Log-MAP или его под-оптимальные варианты для декодирования компонентных кодов, приведена на рисунке 1.9.

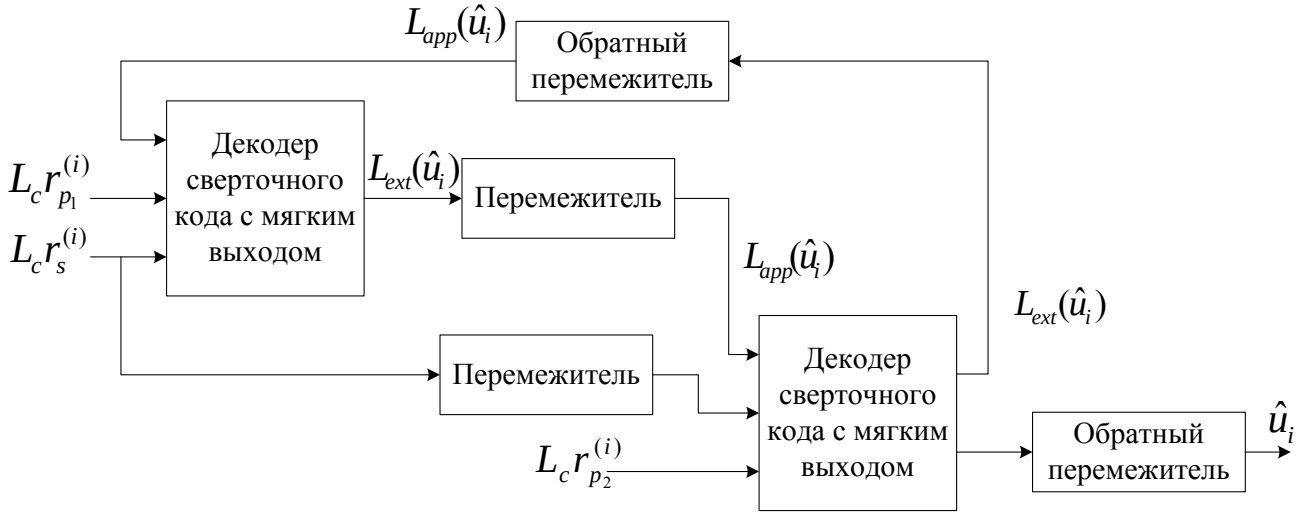

Рисунок 1.9 – Схема итеративного декодера турбокода

В начале декодирования априорные надежности на входе первого декодера равны нулю. В свою очередь первый компонентный декодер вычисляет внешние надежности, которые интерпретируются как априорные на входе второго декодера на следующей полу-итерации. Второй декодер использует полученную информацию для более надежного декодирования бит и вычисляет новую последовательность внешних надежностей для последующего итерирования. Таким

образом итеративно изменяются и корректируются априорные надежности бит, что позволяет более точно декодировать принятое слово.

В зависимости от используемого алгоритма, который был выбран для декодирования компонентного кода, а также от параметров канала, число итераций декодера может быть различным. В момент, когда дальнейшее итерирование декодера уже не приводит к изменению знака выходных надежностей бит, говорят что декодер сошелся. В работах [20, 40] показано, что турбо-декодер, который использует механизм обмена внешними вероятностями, всегда сходится к некоторому стационарному состоянию и заканчивает свою работу. Обычно достаточно 8-16 итераций для того, чтобы декодер сошелся [35]. Для оценки того, сошелся декодер или нет, и для досрочного завершения итерирования пользуются различными критериями остановки [23–25, 41]. Их можно разделить на три группы:

1. Критерии, использующие мягкие решения компонентных декодеров. К ним можно отнести взаимную энтропию внешних надежностей, частоту изменения последних на различных итерациях, минимальное и среднее значение вычисленных надежностей [25, 41].
2. Критерии, использующие жесткие решения компонентных декодеров. В данном случае учитываются декодированные последовательности на различных итерациях [23].
3. Критерии, основанные на дополнительном знании об информационном слове. Часто на практике для проверки корректности декодирования используют CRC, которая также может быть использована для досрочной остановки турбо-декодера [24].

Отдельно рассматривают способы ускорения схождения итеративного декодера для уменьшения задержки декодирования. На схеме, которая представлена на рисунке 1.9, компонентные декодеры последовательно обмениваются внешними надежностями, однако возможны и другие схемы обмена. В работах [26, 27] рассмотрены параллельная и посимвольная схемы. В случае па-

раллельного обмена два компонентных декодера работают одновременно, после чего обмениваются внешними вероятностями. Посимвольная схема является улучшением параллельной и предлагает незамедлительно передавать вновь вычисленную внешнюю вероятность каждого компонентного декодера. Таким образом, в обоих подходах каждый декодер сверточного когда может использовать более актуальную информацию для расчета мягких решений. Как следствие, такие декодеры позволяют уменьшить общее число необходимых итераций для декодирования, однако выигрыша по P_b практически не дают [27,42]. На рисунке 1.10 проиллюстрирован принцип работы обоих подходов.

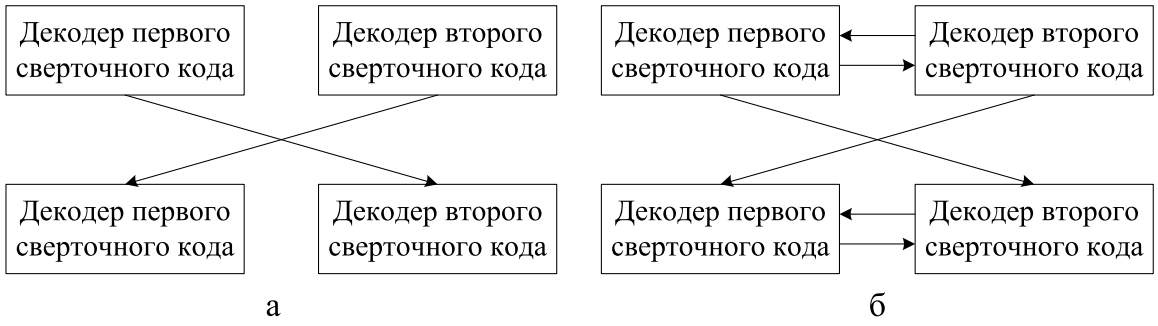

Рисунок 1.10 – Параллельная (а) и посимвольная (б) схемы обмена внешними надежностями

Как видно из вышесказанного, для декодирования турбокода может быть использован любой декодер сверточного кода с мягким выходом. Как будет показано ниже по сравнению с алгоритмом Log-MAP, Max-Log-MAP и SOVA показывают худшую скорость схождения турбо-декодера и худшие корректирующие показатели. Это связано с тем, что вычисленные приблизительно значения внешних надежностей являются завышенными [43,44]. Для того, чтобы исправить это используют взвешивание внешних надежностей, т.е. значение $L_{ext}(\hat{u}_i)$ умножают на некоторый положительный скалирующий коэффициент $w_i < 1$. Алгоритм Max-Log-MAP, для которого производится взвешивание внешних надежностей называют Scaled Max-Log-MAP.

1.5 Результаты моделирования в канале с АБГШ

Для сравнения описанных алгоритмов декодирования турбокода была использована модель системы представленная на рисунке 1.5. В качестве турбокода был рассмотрен турбокод стандарта 3GPP LTE [45], который состоит их двух сверточных кодов с 8-мью состояниями и одним внутренним перемежителем. Порождающий полином компонентного кода выглядит как

$$G(D) = \left[1, \frac{g_1(D)}{g_0(D)}\right],$$

где

$$g_0(D) = 1 + D^2 + D^3,$$

$$g_1(D) = 1 + D + D^3.$$

Перемежитель задается как квадратичный полином перестановки следующего вида

$$\Pi(i) = (f_1 \cdot i + f_2 \cdot i^2) \mathrm{mod}\ K,$$

где $\Pi(i)$ номер коэффициента i после перемежения, а f_1 и f_2 коэффициенты, зависящие от длины информационного слова и определенные в [45]. После окончания кодирования компонентные коды замыкаются в нулевое состояние.

Сравнение алгоритмов декодирования производилось по двум метрикам - вероятности ошибки на бит информационного слова (Bit Error Rate, BER) и вероятности ошибки при декодировании всего информационного слова (Frame Error Rate, FER). Значения BER и FER рассчитывались как

$$\mathrm{BER} = \frac{\text{Общее число ошибочных бит}}{\text{Общее число переданных бит}},$$

$$\mathrm{FER} = \frac{\text{Общее число ошибочных пакетов}}{\text{Общее число переданных пакетов}}.$$

Для сравнения алгоритмов использовалось отношение сигнал/шум на информационный бит E_b/N_0 (дБ).Поскольку в качестве модуляции была использована двоичная фазовая манипуляция, то значение E_s было принято за единицу.

Результаты сравнения описанных выше алгоритмов декодирования турбокода в канале с АБГШ показаны на рисунке 1.11. Моделирование было проведено для 8 итераций декодера. Взвешивающий коэффициент алгоритма Scaled Max-Log-MAP был найден с помощью моделирования и уставлен равным 0,75 для всех символов. Как видно на представленных графиках, алгоритм Scaled Max-Log-MAP со взвешивающим коэффициентом 0,75 работает практически также хорошо, как и Log-MAP, при том что Scaled Max-Log-MAP обладает всеми плюсами алгоритма Max-Log-MAP, т.е. прост, не требует вычисления экспонент и знания дисперсии канала. Алгоритмы Max-Log-MAP и SOVA сильно проигрывают декодеру Log-MAP.

Также рассмотрена зависимость производительности различных декодеров от количества итераций, результаты моделирования показаны на рисунке 1.12. Как видно из приведенных результатов, общей чертой всех описанных алгоритмов является то, что после 4-5 итерации рост производительности замедляется и уже между 7 и 8 итерациями рост практически не заметен. Следовательно можно сделать вывод, что 8-16 итераций достаточно для декодирования турбокодов.

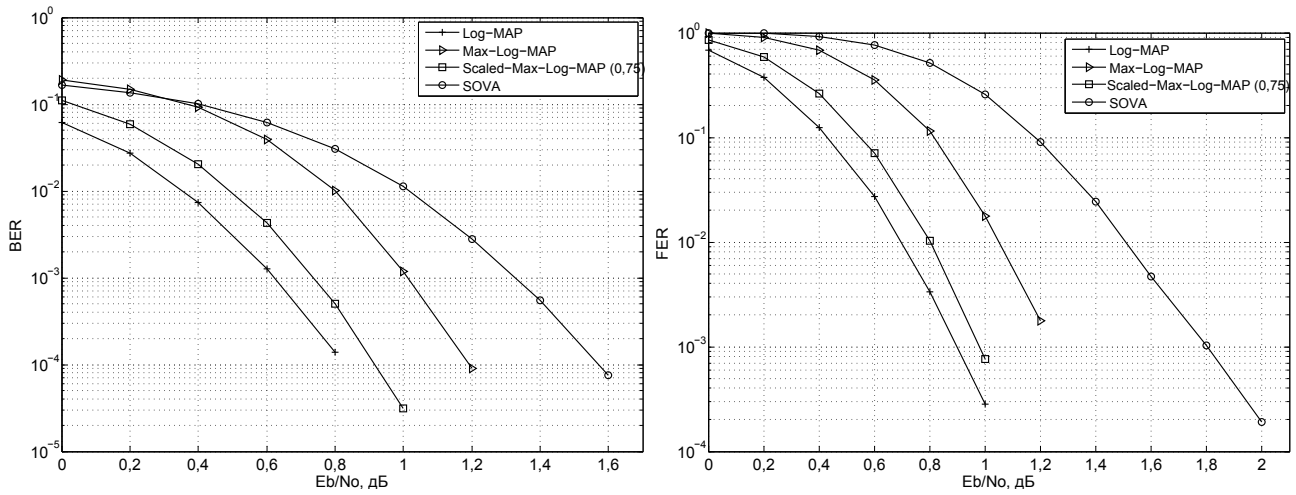

Рисунок 1.11 – Сравнение турбо-декодеров с различными компонентными декодерами ($k = 1024$)

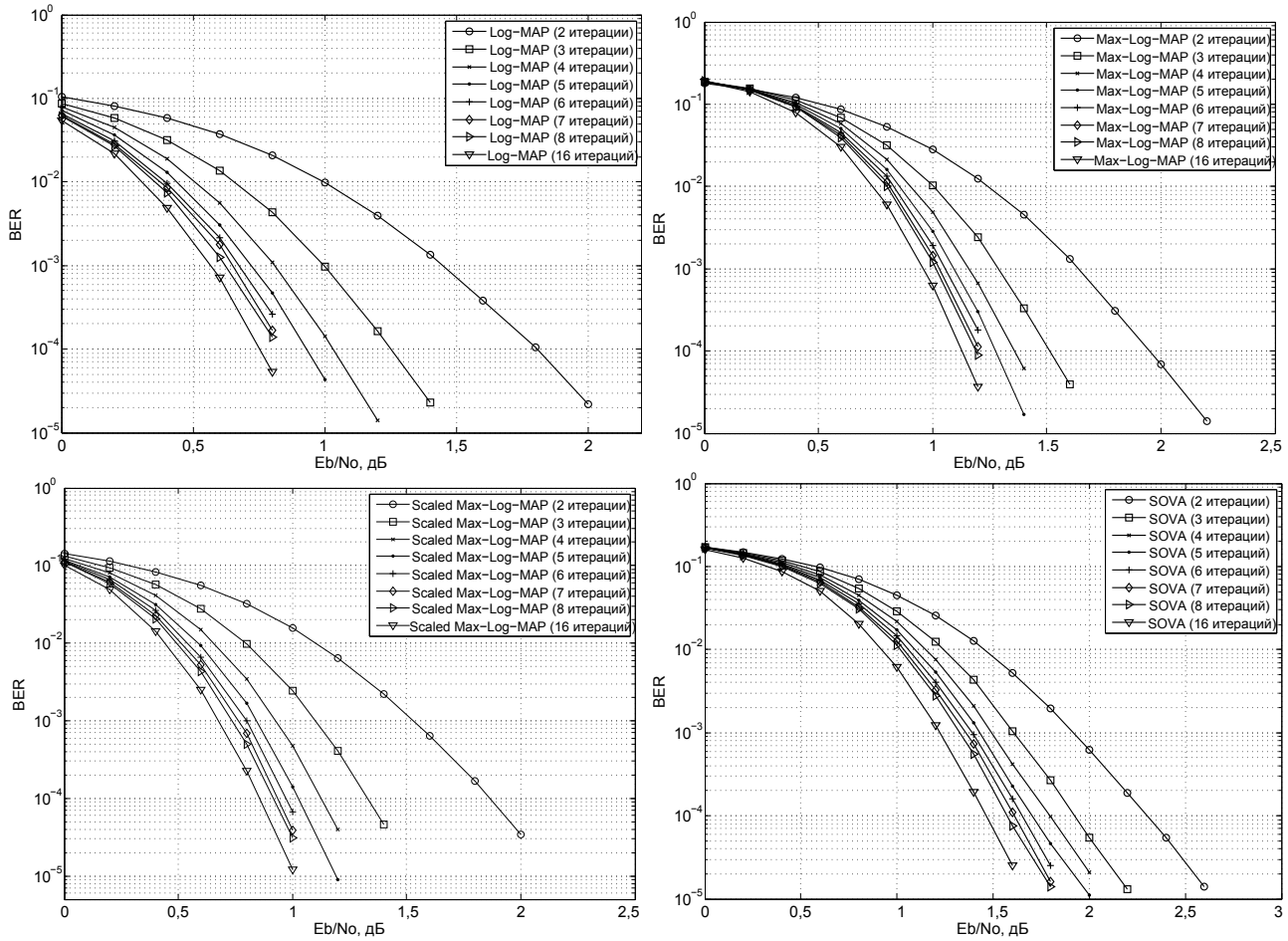

Рисунок 1.12 – Сравнение турбо-декодеров с различным числом итераций ($k = 1024$)

1.6 Заключение и выводы по разделу

В разделе рассмотрены принципы построения турбокода, а также дано описание сверточных кодов и различных видов перемежителей, как основных компонент турбокода. Описан принцип итеративного декодирования турбокодов на основе мягких декодеров сверточных кодов. Т.к. на данный момент итеративный способ является основным методом декодирования турбокодов, приведены результаты моделирования на основе различных мягких декодеров сверточных кодов и рассмотрена их производительность при различном числе итераций.

Материалы раздела носят, в основном, обзорный характер, но позволяют сделать вывод о том, что алгоритм Scaled Max-Log-MAP является лучшим выбором при реализации в практических системах. Также, полученные результаты

говорят о том, что после 8 итераций декодера рост производительности значительно замедляется для всех рассмотренных алгоритмов. По этой причине в дальнейших разделах при моделировании используется алгоритм Scaled Max-Log-MAP с 8 итерациями и взвешивающим коэффициентом 0,75.

Итеративный декодер является подоптимальным декодером, однако декодирование по максимуму правдоподобия турбокода сложно и не реализуемо, поэтому итеративный способ остается единственным и лучшим методом декодирования на данный момент. Но можно говорить о том, что возможно дальнейшее улучшение параметров декодирования турбокода, а т.к. степень распространения последних велика, то эта задача является актуальной. Этому и посвящены последующие разделы данной диссертации.

2 Списочный подход к декодированию турбокодов

2.1 Вводные замечания

Существует несколько подходов улучшения производительности турбокода. Первый подход связан непосредственно с проектированием кода, который заключается в выборе компонентных кодов и перемежителя. Второй подход, это оптимизация декодера, и это единственный способ, когда речь идет о повышения качества приема в существующих стандартах и системах.

Одним из основных путей улучшения параметров декодирования турбокодов является списочное декодирование. Можно выделить две основные схемы построения списочного декодера. Первая схема списочного декодирования использует один или два списочных декодера компонентных кодов, основанных на алгоритме Витерби [46, 47]. Анализ показывает, что выигрыш, который достигается при использовании этой схемы, виден лишь в области насыщения вероятности ошибки на бит или пакет , в то время как выигрыша в области спада вероятности ошибки, наиболее интересной с практической точки зрения, практически нет. Вторая схема основана на проверочной матрице турбокода, которая может быть выписана для определенной длины перемежителя. В этом случае для его декодирования может быть использован алгоритм распространения доверия [48]. Однако, по причинам, описанным ниже, данный подход практически не дает выигрыша применительно к турбокодам.

В данном разделе предложен другой подход к списочному декодированию турбокодов, основным элементом которого является декодер сверточного кода со списком мягких решений. Каждый мягкий выход является последовательностью априорных вероятностей и инициализирует независимый турбо-декодер. Идея подхода состоит в том, чтобы разные декодеры сошлись к разным информационным словам, среди которых и выбирается правильное.

В разделе 2.2 описаны и проанализированы существующие способы списочного декодирования турбокодов. В разделе 2.3 предложен новый алгоритм списочного декодирования. Раздел 2.4 посвящен анализу предложенного метода.

2.2 Обзор существующих методов списочного декодирования турбокодов

2.2.1 Списочное декодирование линейных кодов

Впервые понятие списочного декодирования ввел в 1957 П. Элиас [49]. Рассмотрим некоторый линейный код C с минимальным расстоянием d и обозначим как $C(\mathbf{u})$ кодовое слово данного кода, соответствующее информационному слову $\mathbf{u}$. Если при передаче в кодовом слове $\mathrm{C}(\mathbf{u})$ произошло τ ошибок, было принято слово $\mathbf{r}$ и число τ меньше чем $d/2$ (рисунок 2.1), то слово будет однозначно декодировано. Если τ больше чем $d/2$, то декодировать такое слово не удастся, но можно найти список слов, состоящий из конечного числа элементов, в котором будет содержаться передаваемое слово (рисунок 2.1). Таким образом, задача списочного декодирования заключается в нахождении списка сообщений $\mathbf{U} = (\mathbf{u_1}, \mathbf{u_2}, \ldots)$ таких, что расстояние Хэмминга между $\mathbf{r}$ и $C(\mathbf{U})$ будет не больше e, где e число ошибок, которые необходимо исправить. Если в списке имеется передаваемое кодовое слово, то декодирование считается успешным. Списочное декодирование позволяет исправлять ошибки в случае, когда $e \gg d/2$. И даже в случае, когда размерность списка мала, списочный декодер позволяет исправляет больше чем $d/2$ ошибок.

Одним из критериев выбора правильного слова может быть критерий максимума правдоподобия, в этом случае выбирается слово, которое ближе всего к принятому из канала слову. В случае канала с АБГШ выбирается последовательность ближайшая в смысле Евклидова расстояния. Также на приемнике может

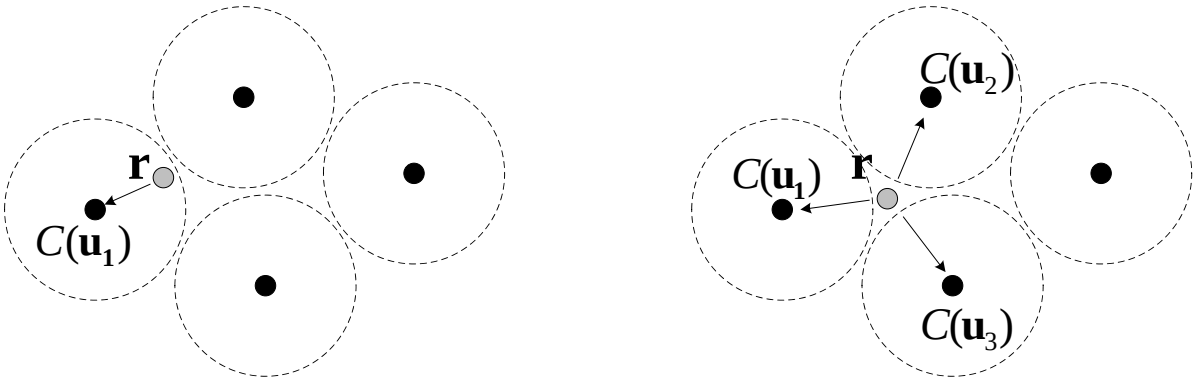

Рисунок 2.1 – Слева: однозначное декодирование принятого слова $\mathbf{r}$ в случае, когда $\tau < d/2$. Справа: списочное декодирование принятого слова $\mathbf{r}$ в случае, когда $\tau > d/2$ и передавалось сообщение $C(\mathbf{u_1})$. Список будет содержать сообщения $C(\mathbf{u_1})$, $C(\mathbf{u_2})$, $C(\mathbf{u_3})$.

иметься дополнительная информация, которая будет использована для выбора правильного слова. На практике часто параллельно с канальным кодированием используется проверка CRC, и если в списке есть слово, которое удовлетворяет проверке, то оно считается правильным.

Долгое время списочное декодирование было невостребованным. Одной из причин являлось то, что были не известны эффективные списочные декодеры для существовавших кодов. Другая причина заключалась в отсутствии технической базы, на которой можно было реализовать сложные списочные декодеры. В последнее время интерес к ним растет, что связано с появлением новых классов кодов и появлением высокоскоростных платформ, на которых декодеры могут быть реализованы. К кодам, для которых существуют эффективные алгоритмы списочного декодирования можно отнести: коды Рида-Соломона [50, 51], Рида-Маллера [50, 51] и сверточные коды [29]. Значительный интерес представляет списочное декодирование турбокодов.

Можно выделить два подхода списочного декодирования турбокодов. Основной чертой первого является то, что в нем используется списочное декодирование сверточных кодов с жестким выходом [46, 47, 52]. В качестве последнего используют списочный декодер Витерби [53], выходом которого является по-

следовательность декодированных слов, из которых в дальнейшем выбирается правильное информационное слово. Т.к. турбо-декодер состоит из двух декодеров сверточного кода, то список может строится на любом из них и в разные моменты итерирования турбо-декодера.

В случае второго подхода списочного декодирования турбокода выписывается проверочная матрица турбокода, после чего с помощью алгоритма распространения доверия выполняется декодирование [54]. Основным фактором в данном подходе, влияющим на производительность декодера, является разреженность проверочной матрицы и наличие в ней циклов короткой длины [48,55]. Ниже даны описание и анализ обоих алгоритмов.

Отдельно стоить отметить подход к списочному декодированию турбокода, при котором для улучшения производительности декодера и нахождения списка слов используется декодирование проверки CRC [56], которая часто добавляется в практических системах. Однако в диссертационной работе такой подход не рассматривается по причинам, которые будут описаны в третьем разделе.

2.2.2 Списочный декодер Витерби

Различают два способа реализации списочного декодера Витерби - параллельный (Parallel List Viterbi Algorithm, PLVA) и последовательный (Serial List Viterbi Algorithm, SLVA) [53]. Как следует из названия, PLVA находит все элементы списка за один проход по решетке сверточного кода, в то время как SLVA для нахождения i-ого слова предварительно рассчитывает $i-1$ предыдущую последовательность. Найденные декодером последовательности или пути, количество которых обозначим как L, являются наиболее вероятными на решетке сверточного кода [53].

Алгоритм PLVA работает по тому же принципу, что и алгоритм Витерби. Отличие заключается лишь в том, что в каждом состоянии решетки хранится информация не об одном лучшем пути, а об L выживших путях. В конце де-

кодирования параллельно выполняется L обратных проходов по решетке, что соответственно дает L информационных слов.

В общем виде алгоритм PLVA можно описать следующим образом:

1. Для каждого состояния q найти метрики входящих путей по формулам

$$M_i^j(q) = M_{i-1}^j(p_1) + M_i(\boldsymbol{r_i}, \hat{\boldsymbol{v}}_{\boldsymbol{i}}^{(\boldsymbol{p_1},\boldsymbol{q})}), \tag{2.1}$$

   и

$$M_i^j(q) = M_{i-1}^j(p_2) + M_i(\boldsymbol{r_i}, \hat{\boldsymbol{v}}_{\boldsymbol{i}}^{(\boldsymbol{p_2},\boldsymbol{q})}), \tag{2.2}$$

   где $j \in \{0, \ldots, L-1\}$ - индекс пути.
2. Из $2L$ путей, входящих в состояние q, выбрать L путей с максимальной метрикой.
3. Сохранить соответствующие состояния p, из которых пришли выжившие L путей с максимальной метрикой.
4. Повторить шаги 1-3 для всех состояний решетки, после чего в каждом состоянии решетки будет содержаться информация об L выживших путях.
5. Если известно, что решетка замыкается в состоянии 0:
   (a) Из конченого состояния выполнить обратный проход по решетке для каждого из L выживших путей и найти соответствующие битовые последовательности.
6. Если конечное состояние пути на решетке не известно:
   (a) Найти конечные состояния L путей с максимальными метриками и из них выполнить обратный проход по решетке для восстановления битовых последовательностей.

Алгоритм SLVA находит элементы списка по одному и в первую очередь находит лучший путь [57]. При этом, как и в алгоритме SOVA, сохраняются разности частичных метрик лучшего пути и отброшенных в каждом состоянии. В том состоянии, где рассчитанная разность минимальна, сходятся первый и второй лучшие пути. Следовательно, для получения второго лучшего пути, необходимо выполнить обратный проход по решетке из этого состояния. Для нового

найденного пути также сохраняются разности частичных метрик. Для нахождения третьего лучшего пути находится состояние с минимальным значением разниц двух лучших путей и из него выполняется обратный проход для нахождения третьей последовательности. Эта операция повторяется пока не будут найдены все элементы списка.

2.2.3 Списочное декодирование турбокода на основе списочного декодера сверточного кода

Впервые использовать списочный декодер Витерби для декодирования турбокода предложил Д. С. Садовски [47]. В своем алгоритме Садовски с помощью алгоритма SLVA последовательно находит информационные слова для первого и второго компонентных кодов, и попарно сравнивает их между собой, предварительно переставив биты для последовательностей с выхода второго компонентного кода в соответствии с перемежителем турбокода. Когда какой-то элемент списка для первого сверточного кода равен элементу списка для второго, то данная последовательность считается декодированным словом и декодирование заканчивается. Таким образом, критерием остановки декодирования является равенство двух элементов в списках первого и второго компонентных кодов. Однако, как пишет сам автор, такой подход очень сложен и не реализуем при низких отношениях сигнал-шум, а также при длине информационного слова больше 40 бит.

Другой подход предложил в своей работе К. Р. Нарайанан [46], где он использует PLVA после того, как турбо-декодер выполнил все итерации декодирования. Схема декодера показана на рисунке 2.2. Выбор такого подхода оправдан тем, что на первых итерациях турбо-декодер исправляет большое число ошибок, а генерация списка после последние итерации позволяет исправить ошибки, с которыми обычный турбо-декодер не справился.

Поскольку на списочный декодер Витерби при турбо-декодировании пода-

ются априорные надежности бит, то они также должны быть использованы при вычислении метрик переходов на решетке сверточного кода. В данном случае метрика перехода будет вычисляться по формуле 1.35 [46].

Рисунок 2.2 – Списочное декодирование турбокода с использованием списочного декодера Витерби

Недостатком такого подхода является то, что он дает выигрыш лишь в области насыщения вероятности ошибки, но не дает выигрыша в области ее спада, наиболее интересной с практической точки зрения. Также с ростом длины информационного слова производительность такого декодера резко падает, при этом увеличение размерности списка не дает улучшения. Это наглядно продемонстрировано на рисунке 2.3. Как видно из результатов моделирования, подход Нарайанана дает небольшой выигрыш на малых длинах информационного слова, в то время как на больших длинах выигрыша получить не удается, в том числе и при увеличении размерности списка.

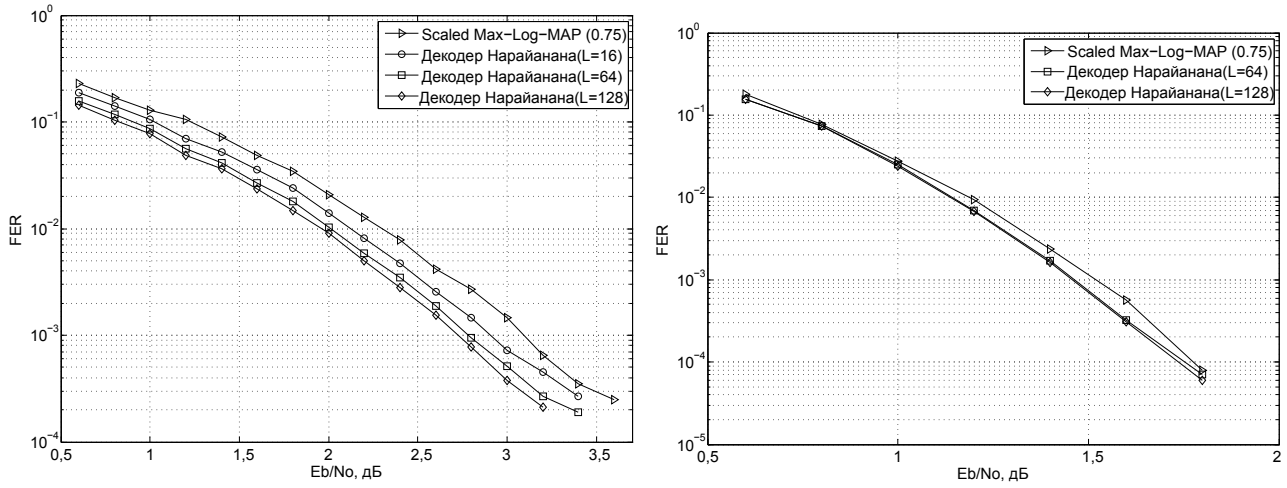

Рисунок 2.3 – Результаты моделирования списочного декодера Нарайанана для $k = 40$ (слева) и $k = 320$ (справа)

2.2.4 Списочное декодирование турбокода на основе недвоичного алгоритма распространения доверия

Т.к. длина перемежителя турбокода имеет ограниченную размерность, то для него может быть выписана проверочная матрица, составными элементами которой являются проверочные матрицы компонентных кодов. В качестве примера рассмотрим турбокод стандарта 3GPP LTE, который был описан в первом разделе работы. Проверочная матрица его компонентного кода может быть выписана как

$$H_{cc}^{T}(D) = \begin{bmatrix} g_1(D) \\ g_0(D) \end{bmatrix} = \begin{bmatrix} 1 + D + D^3 \\ 1 + D^2 + D^3 \end{bmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 & & \\ 1 & 0 & 1 & 1 & & \\ & 1 & 1 & 0 & 1 & \\ & 1 & 0 & 1 & 1 & \\ & & \ddots & \ddots & \ddots & \ddots \end{pmatrix}. \quad (2.3)$$

Так в кодовом слове первыми стоят информационные биты, после которых следуют проверочные, то строки матрицы $H_{cc}^{T}(D)$ следует переставить в соответствии с порядком следования бит. Тогда проверочная матрица компонентного кода примет следующий вид

$$H_{cc}^{T'} = \begin{pmatrix} 1 & 1 & 0 & 1 & & & \\ & 1 & 1 & 0 & 1 & & \\ & & 1 & 1 & 0 & 1 & \\ & & & \ddots & \ddots & \ddots & \ddots \\ 1 & 0 & 1 & 1 & & & \\ & 1 & 0 & 1 & 1 & & \\ & & 1 & 0 & 1 & 1 & \\ & & & \ddots & \ddots & \ddots & \ddots \end{pmatrix} = \begin{pmatrix} H_{cc}^{i} \\ H_{cc}^{r} \end{pmatrix}, \quad (2.4)$$

где H_{cc}^{i} и H_{cc}^{r} матрицы, на которые умножаются информационная и проверочная части кодового слова, соответственно. С учетом введенных обозначений прове-

рочная матрица турбо кода будет выглядеть как

$$H = \begin{bmatrix} H_{cc}^{i} & H_{cc}^{r} & 0 \\ H_{cc}^{i} P^{T} & 0 & H_{cc}^{r} \end{bmatrix}, \tag{2.5}$$

где P^T перестановочная матрица, которая соответствует перемежителю турбокода.

Проверочная матрица H является разреженной, т.е. число нулей в матрице много больше числа единиц. По это причине, как и в случае LDPC кодов, для декодирования турбокода может быть использован алгоритм двоичного распространения доверия [48]. Однако для того, чтобы LDPC код хорошо декодировался с помощью итеративного декодера необходимо, чтобы в проверочной матрице было как можно меньше циклов короткой длины [55]. В случае турбокодов проверочная матрица обладает большим числом циклов короткой длины, что негативно сказывается на производительности алгоритма. В работах [58, 59] показано, что алгоритм двоичного распространения доверия значительно проигрывает обычному Log-MAP алгоритму.

Для уменьшения количества циклов короткой длины Деклерк в работах [54, 60] предложил группировать блоки проверочной матрицы. Иллюстрация процесса группировки или кластеризации в блоки размера 2×2 показана на рисунке 2.4. В результате количество коротких циклов уменьшается, однако для декодирования такой недвоичной матрицы необходимо использовать недвоичный алгоритм распространения доверия, что значительно усложняет декодер. Основной задачей в таком подходе является правильная обработка проверочной матрица и правильное разбиение последней на блоки [60]. От размера блока зависит не только количество циклов в проверочной матрице, но и сложность алгоритма декодирования. Чем больше размер блоков, тем сложнее становится декодер.

Алгоритмы двоичного и недвоичного распространения доверия сходятся после 30-40 итераций. Но в случае недвоичного алгоритма, проверка корректно-

сти декодирования должна выполняться на каждой итерации и после того, как на одной из итераций было получено правильное кодовое слово, необходимо закончить процесс декодирования. Это связано с тем, что дальнейшее итерирование может привести к неверному декодированию и будет получено ошибочное слово [54]. Данное свойство приводит к дополнительному увеличению сложности декодера.

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \longrightarrow H_{cl} = \begin{pmatrix} 1 & 0 & a & b \\ 0 & 1 & b & a \end{pmatrix}$$

Рисунок 2.4 – Иллюстрация процесса кластеризации проверочной матрицы кода

Как показывает анализ в работе Деклерка [54], описанный выше алгоритм незначительно проигрывает алгоритму Log-MAP. Но т.к. от способа обработки проверочной матрицы зависит успешность декодирования, то для улучшения производительности декодера автор предлагает использовать списочный подход. Для генерации списка декодированных слов формируется несколько проверочных матриц с использованием различных способов обработки, на которых в дальнейшем независимо запускаются алгоритмы недвоичного распространения доверия. В результате получается список кодовых слов, из которых выбирается правильное. За счет такого подхода автор получает выигрыш до 0,15 дБ для списка длины 5 и 100 итераций декодера. Но такой результат можно получить лишь на определенных турбокодах, для которых проверочная матрица обладает малым числом циклов. Для других кодов метод может не дать прироста производительности. Это видно в случае декодирования турбокода стандарта 3GPP LTE для 100 итераций и 5 параллельных декодеров (рисунок 2.5). Если на длине информационного слова равного 40 бит выигрыш равен 0,25 дБ, то уже на 64 битах выигрыш составляет порядка 0,1 дБ.

Помимо описанных выше недостатков подхода, очевидным является то,

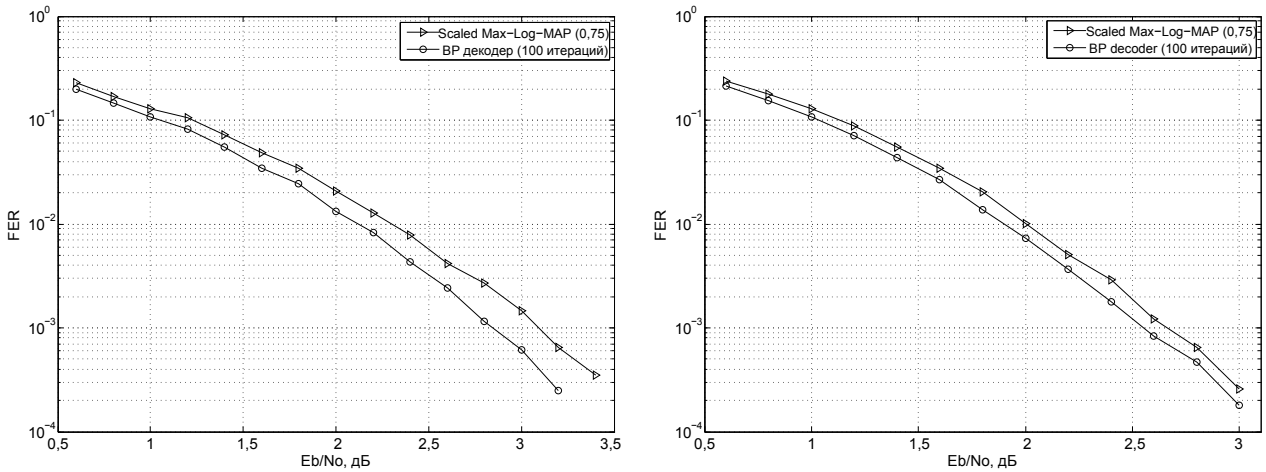

Рисунок 2.5 – Результаты моделирования списочного декодера на основе недвоичного алгоритма распространения доверия для $k = 40$ (слева) и $k = 64$ (справа)

что использование его в реальных системах затрудняется следующими обстоятельством. Поскольку длина перемежителя турбокода может быть различной и зависит от длины информационного слова, то для использования такого декодера необходимо хранить большое число проверочных матриц, и для каждой такой матрицы знать при каком способе группировки декодер даст наилучший результат. Дополнительная сложность обусловлена тем, что задача нахождение способа группировки связана с компьютерным поиском.

2.3 Параллельный алгоритм списочного декодирования турбокода

Анализ существующих методов списочного декодирования показал, что они дают выигрыш лишь в области насыщения вероятности ошибки, либо на маленьких длинах информационных слов. В этом разделе предложен другой подход к списочному декодированию - параллельное списочное декодирование. Отличительной чертой подхода является то, что вначале процесса турбодекодирования генерируется список априорных вероятностей, которые, в свою очередь, подаются на вход независимых турбо-декодеров (рисунок 2.6). По ана-

логии с алгоритмом недвоичного распространения доверия, цель данного метода состоит в том, чтобы разные декодеры сошлись к разным информационным словам, среди которых далее выбирается правильное. В то время как в алгоритме, предложенном Деклерком, различные процессы декодирования инициализируются различными недвоичными матрицами, в данном подходе декодеры инициализируются различными априорными последовательностями.

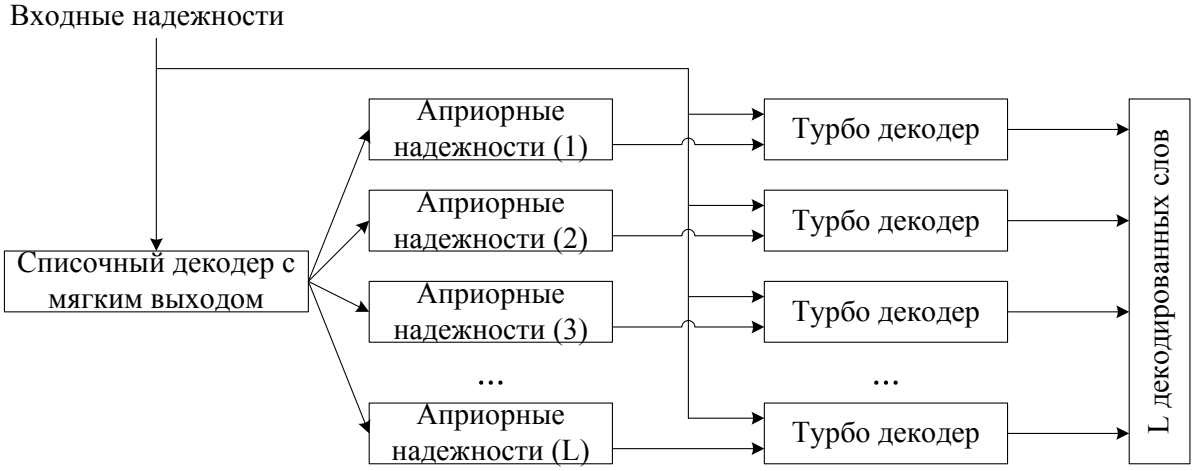

Рисунок 2.6 – Схема предложенного списочного декодера турбокода

При использовании обычного турбо-декодера априорные надежности на входе первого декодера равны 0 и все биты считаются равновероятными. В рассматриваемом подходе начальные априорные надежности инициализируются некоторыми значениями, что приводит к различному поведению турбо-декодеров. Основной задачей предложенного подхода списочного декодирования является нахождение списка мягких решений. При итерировании турбо-декодера эти значения определяют дальнейшее поведение декодера. Если начальные значения заданы произвольно, то дальнейшее поведение декодера сложно предугадать. Поэтому для генерации списка мягких решений ниже предлагается использовать один из компонентных кодов и надежности принятых из канала символов.

2.3.1 Списочный декодер сверточного кода с мягким выходом

Для нахождения списка априорных надежностей предлагается алгоритм, который представляет собой комбинацию декодеров PLVA и Log-MAP. Выходом данного алгоритма является L последовательностей априорных надежностей или мягких решений.

Рассмотрим работу алгоритма на примере, представленном на рисунке 2.7. В начале с помощью алгоритма PLVA находятся L путей на решетке компонентного кода. Первая последовательность списка соответствует внешним надежностям обычного алгоритма Log-MAP (рисунок 2.7 (А)). Для получения второй последовательности делается предположение, что первый путь в решетке является ошибочным, в то время как второй путь верен. Поэтому из решетки сверточного кода выкалываются ребра, которые принадлежат первому пути, но не принадлежат второму (рисунок 2.7 (Б)). На получившейся решетке сверточного кода с выколотыми ребрами выполняется алгоритм Log-MAP, внешние надежности которого являются второй последовательностью списка. Для получения третьего элемента списка делается предположение, что первые два пути не верны и из полной решетки сверточного кода выкалываются ребра, которые принадлежать первому и второму путям, но не принадлежат третьему (рисунок 2.7 (В)). В получившейся решетке также выполняется алгоритма Log-MAP и рассчитывается третий элемент списка. Эта же процедура выполняется для нахождения любого другого элемента списка.

Log-MAP алгоритм, выполненный на решетке с выколотыми ребрами, рассчитывает априорные надежности без вклада предположительно ошибочных переходов. Т.к. эти переходы принадлежали лучшим путям и имеют большой вес, то они сильно влияют на итоговые априорные значения. Таким образом, при расчете апостериорных вероятностей бит на редуцированной решетке делается попытка избавиться от этого влияния. В результате получаются другие априорные значения. По сути, алгоритм вычисляет мягкие решения для различных

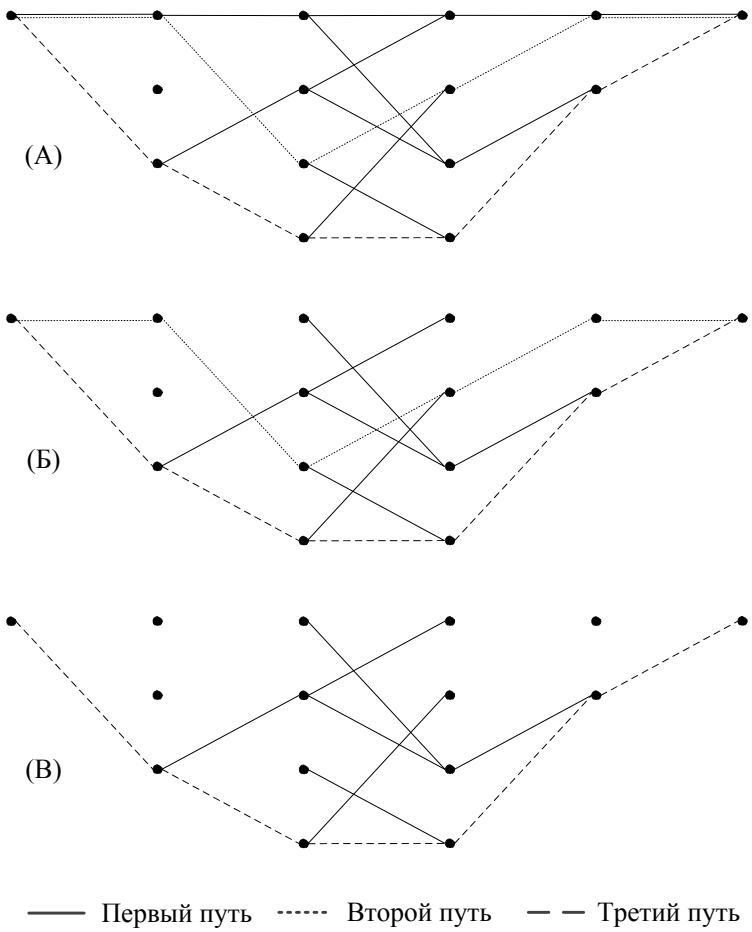

Рисунок 2.7 – Иллюстрация работы списочного декодера сверточного кода с мягким выходом

решеток, из которых удалены ложные переходы, т.е. мягкие выходы можно считать уточненными при переходе от одного элемента списка к другому. Ребра, которые принадлежат нескольким путям, имеют большую надежность. Те же ребра, которые, например, принадлежат первому пути, но не принадлежат последующим, могут быть ошибочными и, следовательно, менее надежными. Удаляя из решетки такие ребра, которые будут рассмотрены в более ранних элементах списка, алгоритм пытается рассматривать менее вероятные альтернативные решения. Также можно рассчитывать на то, что соответствующие турбо-процессы сойдутся к различным информационным словам.

В общем виде алгоритм выглядит следующим образом:

1. Найдем L самых вероятных путей в решетке сверточного кода с помощью списочного алгоритма Витерби.

2. Первый элемент списка равен внешним надежностям Log-MAP декодера.
3. Обозначим как B_i, где $i = 1, \ldots, L$, множество всех ребер в решетке, которые принадлежат пути i. Для нахождения j-ого мягкого выхода:
   (a) Выколем из решетки все ребра, которые принадлежат множествам B_i, для $i = 1, \ldots, j-1$, и вставим обратно ребра, которые принадлежат множеству B_j. Т.е. уберем из решетки все ребра из множества $\left(\cup_{i=1}^{j-1} B_i - \cup_{i=1}^{j-1}(B_i \cap B_j)\right)$.
   (b) Выполним Log-MAP алгоритм на решетке с выколотыми ребрами. Рассчитанные внешние надежности являются j-ым элементом списка.
   (c) Повторим шаги a-b для получения всех элементов списка.

Стоит отметить, что в для расчета мягких решений может использоваться не только Log-MAP алгоритм, но и его подоптимальные варианты Max-Log-MAP и Scaled Max-Log-MAP.

Утверждение 2.1. *Все мягкие выходы, рассчитанные списочным декодером сверточного кода с мягким выходом для кода с замыканием решетки в нуле, являются различными.*

Доказательство. Если в качестве алгоритма расчета мягких решений используется алгоритм Log-MAP, то данное утверждение является очевидным. Удаление даже одного ребра (p, q) в i-ой секции решетки приведет к тому, что значения $A_{i+1}(q)$ для состояний в секциях от $i+1$ до k и значений $B_i(p)$ для состояний в секциях от 1 до i будут изменены. Это, в свою очередь, приведет к изменению значений внешних надежностей.

Рассмотрим случай, когда для расчета мягких значений используется алгоритм Max-Log-MAP, и рассчитывается второй элемент списка. Пусть было удалено ребро (p, q) в i-ой секции решетки. Т.к. для расчета значений $A_{i+1}(q)$ используется формула 1.37, тогда будут изменены все значения $A_j()$ для состояний вдоль лучшего пути, от $(i+1)$-ой до последней секции. По этой же причине будут изменены все значения $B_j()$ для состояний вдоль лучшего пути, от 1-ой

до i-ой секции. Таким образом, как минимум надежность первого и последнего бит будут отличаться от списка к списку. Это справедливо и для последующих элементов списка. □

2.3.2 Оконный списочный декодер сверточного кода с мягких выходом

При декодировании сверточных кодов часто используют оконный подход для более эффективной реализации декодера [61–63]. Оконный подход к декодированию сверточных кодов позволяет реализовать декодер с меньшими затратами на память и вычислительные ресурсы. Также данный подход позволяет распараллеливать вычисления.

Можно выделить два подхода оконного декодирования сверточного кода: со скользящим окном и параллельное оконное декодирование. Т.к. алгоритм со скользящим окном в каждый момент времени рассчитывает надежность одного символа и не строит пути на решетке, то он не может быть использован применительно к предложенному алгоритму. Параллельный подход разбивает решетку на окна определенной длины, в каждом из которых независимо, а следовательно и в параллель выполняется декодирование. Ниже рассмотрено использование последнего подхода для расчета списка мягких решений.

Оконный списочный алгоритм строит списки на участках решетки, которые в дальнейшем используются для получения последовательности априорных вероятностей для всего информационного слова. Для вычисления списка в окне используется тот же подход, что и в алгоритме, который был описан выше, однако вместо обычных алгоритмов Витерби и Log-MAP, используются их оконные варианты. Т.к. списки строятся в каждом окне независимо, то это дает возможность ускорить процесс генерации списка для параллельного декодирования.

При рассмотрении оконного декодера Витерби вводят понятие суффикса, т.к. в окне не известны начальные и конечные состояния пути(рисунок 2.8). Известно, что если длина суффикса равна 3-5 длинам кодового ограничения свер-

точного кода, то выжившие пути в конце суффикса, полученные в результате работы алгоритма Витерби с большой вероятностью имеют общее начало в конце окна (рисунок 2.8). Таким образом, суффикс необходим для нахождения конечного состояния в окне, в то время как начальные состояния считаются равновероятными.

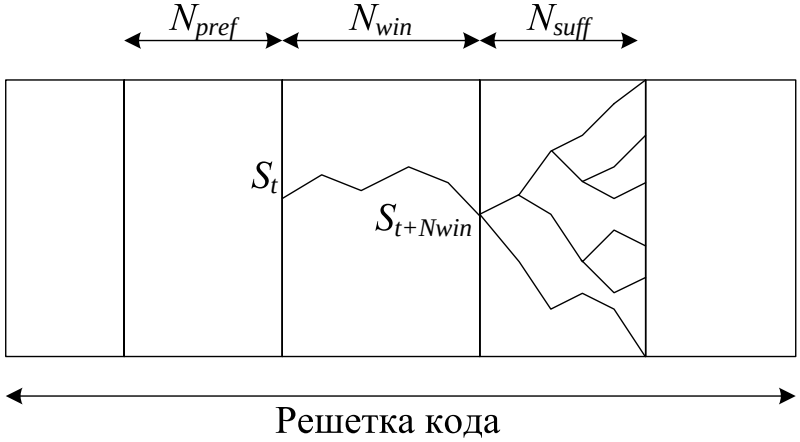

Рисунок 2.8 – Окно в решетке сверточного кода

Обозначим как N_{win} длину окна и N_{suff} длину суффикса. Пусть i номер начальной секции окна. Тогда оконный списочный алгоритм Витерби выглядит следующим образом:

1. Выполним алгоритм Витерби на участке решетки от секции i до $i + N_{win} + N_{suff}$, где начальные и конечные состояния считаются равновероятными.Найденный путь является лучшим в рассматриваемой области решетки.
2. Обозначим начальное и конечное состояния найденного пути в окне как S_i и $S_{i+N_{win}}$, соответственно. Будем считать, что остальные пути в окне являются ответвлениями лучшего пути.
3. Пусть L размер списка в окне. Для нахождения L последовательностей выполним списочный алгоритм Витерби в окне и будем считать, что все начальные состояния равновероятны, в то время как конечное состояние равное $S_{i+N_{win}}$.
4. Из состояния $S_{i+N_{win}}$ выполним L обратных проходов по решетке, после чего будут получены искомые элементы списка.

В результате работы алгоритма мы получим L путей в окне, которые могут иметь произвольное начало, но заканчиваются в состоянии $S_{i+N_{win}}$.

Т.к. в алгоритме Log-MAP выполняется два прохода по решетке для нахождения прямых и обратных метрик, то в оконном варианте помимо суффикса, вводят понятие префикса, которые служат для более корректного вычисления метрик в окне (рисунок 2.8). Т.е. в оконном Log-MAP алгоритме расчет метрик начинается в секции с номером $i - N_{pref}$ и заканчивается в секции с номером $i+N_{win}+N_{suff}$, а начальные и конечные состояния считаются равновероятными. Таким образом, оконный Log-MAP алгоритм сводится к следующим шагам:

1. Инициализируем $A_{i-N_{pref}}(q) = 0$ и $B_{i+N_{win}+N_{suff}}(q) = 0$ для всех $q \in \{0, \ldots, N-1\}$.
2. Вычислим метрики переходов $\Gamma_i(p, q)$ для всех ребер в рассматриваемой области решетки по формуле 1.35.
3. Вычислим метрики состояний $A_i(p)$ в ходе прямого прохода по формуле 1.33.
4. Вычислим метрики состояний $B_{i+1}(q)$ в ходе обратного прохода по формуле 1.34.
5. Вычислим апостериорные вероятности выходных бит в окне решетки по формуле 1.32.

Имея оконные списочный Витерби и Log-MAP алгоритмы, оконный списочный декодер сверточного кода с мягким выходом может быть описан следующим образом:

1. Разбить решетку сверточного кода на Z окон, которые в дальнейшем считаются независимыми.
2. Для каждого окна, чтобы получить список мягких решений, выполнить следующие действия:
   (a) Найти L самых вероятных путей в окне решетки сверточного кода с помощью оконного списочного алгоритма Витерби.

(b) Первый элемент списка равен внешним надежностям оконного Log-MAP декодера.

(c) Обозначим как $B_i^{(z)}$, где $i = 1, \ldots, L$, множество всех ребер в z-ом окне решетки, которые принадлежат пути с номером i. Для нахождения j-ого мягкого выхода:

i. Выколем из решетки все ребра, которые принадлежат множествам $B_i^{(z)}$, для $i = 1, \ldots, j-1$, и вставим обратно ребра, которые принадлежат множествам $B_j^{(z)}$. Т.е. выколем из решетки все ребра из множества $\left(\cup_{i=1}^{j-1} B_i^{(z)} - \cup_{i=1}^{j-1}(B_i^{(z)} \cap B_j^{(z)})\right)$.

ii. Выполним оконный Log-MAP алгоритм в окне решетки с выколотыми ребрами. Рассчитанные внешние надежности являются j-ым элементов списка.

iii. Повторим шаги i-ii для получения всех элементов списка в текущем окне.

Таким образом, решетка сверточного кода разбивается на окна определенной длины, в каждом из которых находится список надежностей. Далее эти списки используются для нахождения внешних вероятностей для всего информационного слова.

2.3.3 Списочное декодирование турбокода с использованием списочного декодера сверточного кода с мягким выходом

Таким образом, для списочного декодирования турбокода возможно два варианта генерации априорных надежностей для независимых турбо-декодеров. В первом случае список генерируется сразу для всей решетки с использованием одного из компонентных кодов. Тогда, в зависимости от того какой компонентный код использовался для нахождения списка, будет зависеть на какой компонентный декодер будут подаваться мягкие решения для дальнейшего турбо-декодирования. В случае канала с независимыми ошибками, таким как АБГШ,

выбор компонентного кода для построения списка не играет роли.

Во втором случае, который актуален для более длинных информационных слов, сперва рассчитываются списки мягких решений с помощью оконного алгоритма, после чего генерируется общий список для инициализации независимых турбо-декодеров. Для того, чтобы найти мягкое решение для всего информационного слова, в каждом окне берется по одному элементу списка и их конкатенация является элементом результирующего списка. Так делается до тех пор пока не будут рассмотрены все возможные комбинации. При разбиении решетки на Z окон с длиной списка L в каждом окне, общий размер списка для всего информационного слова будет равен L^Z. При маленьких значениях Z и L можно перебрать все варианты. В противном случае экспоненциальный рост длины общего списка не позволяет использовать полный перебор, поэтому предложен следующий подход к его уменьшению.

Рассмотрим окно j с длиной списка $L = 2$. Обозначим как $E_{i,j}$ i-ую последовательность списка в j-ом окне. Вычислим Евклидово расстояние между первым и вторым элементами списка

$$\epsilon_j = \sum_{i=1}^{N_{win}} \left(E_{1,j}^i - E_{2,j}^i \right)^2 . \tag{2.6}$$

Делается предположение, что чем больше Евклидово расстояние между двумя векторами, тем с большей вероятностью процессы турбо-декодирования, инициализированные данными элементами списка, сойдутся к различным словам. Таким образом, в каждом окне между элементами списка вычисляется Евклидово расстояние, после чего второй элемент списка удаляется в окнах с наименьшим расстоянием. Данный процесс проиллюстрирован на рисунке 2.9, на котором показано, что за счет удаления вторых элементов списка в первом и третьем окнах общую длину списка удается уменьшить с 8 до 2. Этот же подход можно расширить на значения $L > 2$.

Детальная схема параллельного списочного декодера турбокода показана на рисунке 2.10. Как видно на вход независимых турбо-декодеров подаются вы-

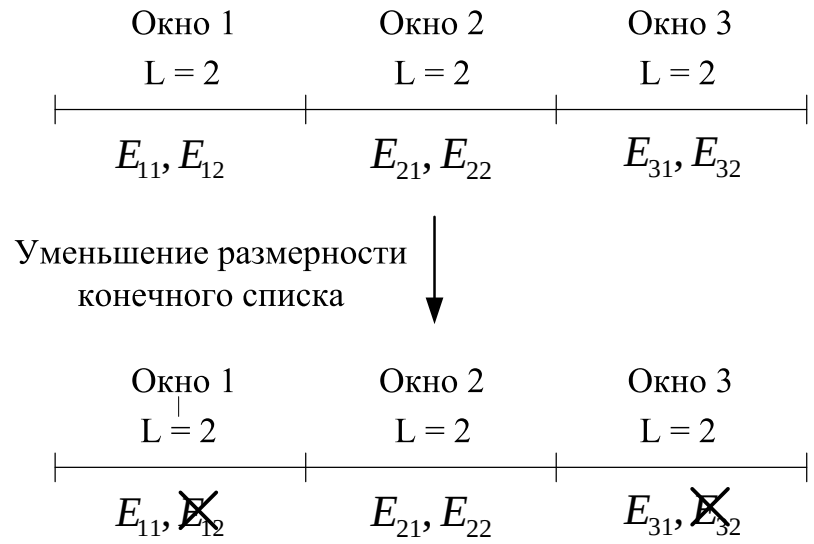

Рисунок 2.9 – Иллюстрация построения итогового списка для оконного подхода

численные априорные последовательности. При этом стоит отметить, что декодирование компонентных кодов происходит по полной решетке кода.

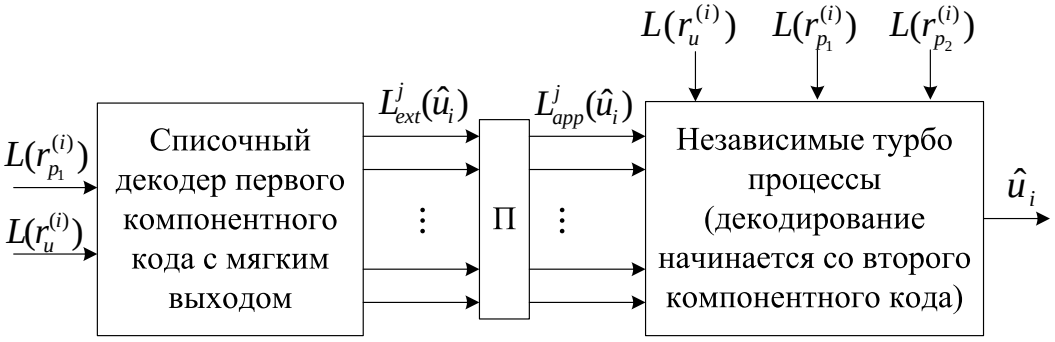

Рисунок 2.10 – Детальная схема параллельного списочного декодера турбокода

Сложность метода пропорциональна количеству турбо-процессов, которые были инициализированы. Сложность одного процесса турбо-декодирования может быть рассчитана исходя из сложности декодирования компонентного кода. В случае, если используется алгоритм Scaled Max-Log-MAP, то в каждой секции решетки необходимо выполнять операции сложения, умножения и нахождения максимума. Обозначим как $C_{SMaxLogMAP}$ число операций для декодирования одной секции решетки, тогда сложность одного турбо-декодера можно вычислить как $2 \cdot C_{SMaxLogMAP} \cdot N_{it} \cdot k$, где N_{it} число итераций декодера. В соответствии с работой [64] значение $C_{SMaxLogMAP}$ складывается из $10N + 11$ операций сложе-

ния, $5N-2$ операций нахождения максимума, 8 операций умножения на ± 1, а также 1 операции умножения на взвешивающий коэффициент.

Для предложенного алгоритма полная сложность зависит от размерности списка и вычисляется как $2 \cdot C_{SMaxLogMAP} \cdot N_{it} \cdot k \cdot L + \mathrm{C}_{LVA}$, где C_{LVA} сложность списочного декодера Витерби. В случае параллельной реализации алгоритма LVA в каждом состоянии решетки необходимо выполнить 2 сложения для вычисления метрик переходов; $2L$ сложений для вычисления частичных метрик входящих в состояние путей; для нахождения L выживших путей необходима сортировка $2L$ значений, что требует приблизительно $2L\log(2L)$ операций. После того, как был достигнут конец решетки сверточного кода необходимо выполнить обратный проход по решетке для восстановления битовых последовательностей.

Задержка декодирования предложенного декодера может быть выражена в числе проходов по решетке компонентного кода. Для алгоритма Scaled Max-Log-MAP необходимо выполнить 3 прохода: первый проход для вычисления метрик $A_{t+1}(q)$, второй - для вычисления метрик $B_t(p)$ и третий для нахождения итоговых надежностей. Общее число проходов по решетке для классического турбо-декодера равно $6N_{it}$. Для предложенного списочного декодера турбокода задержка декодирования возрастает из-за необходимости выполнения списочного декодера Витерби. Если для нахождения списка слов используется алгоритм PLVA, то дополнительно потребуется 2 прохода по решетке компонентного кода, т.е. задержка декодирования станет равной $6N_{it}+2$. В случае параллельной реализации алгоритма LVA задержка увеличивается на $2L$ прохода по решетке.

При практической реализации такого декодера, для декодирования каждого процесса может быть использован уже имеющийся турбо-декодер, реализованный в виде программного или аппаратного модуля. С учетом того, что наиболее популярным декодером турбокода является итеративный декодер на основе алгоритма Log-MAP и существуют эффективные методы оптимизации применительно к конкретным кодам и системам, предложенный алгоритм может быть легко

реализован и внедрен с использованием существующих наработок, быстрых реализаций и оптимизаций на затрачиваемые ресурсы и память. Другим плюсом данного алгоритма является то, что все турбо-процессы могут работать в параллель, при этом вычисление списка также может быть выполнено в параллель за счет использования алгоритма PLVA. Таким образом, задержка такого декодера при аппаратной реализации будет незначительно превышать задержку обычного турбо-декодера. Стоит отметить, что для каждого турбо-процесса для уменьшения общего числа итераций может быть использован алгоритм досрочной остановки итераций.

2.4 Результаты моделирования в канале с АБГШ

Поскольку нельзя спрогнозировать поведение турбо-декодера в зависимости от начальной последовательности априорных надежностей, эффективность выбора таких последовательностей может быть показана лишь с помощью имитационного моделирования. Для оценки производительности списочного декодера использовалась система, представленная на рисунке 1.5. В качестве кода был рассмотрен турбокод стандарта 3GPP LTE, описанный в предыдущем разделе. В стандарт 3GPP LTE на физическом уровне перед кодированием к каждому информационному слову прибавляется проверка CRC длиной 24 бита для обнаружения ошибки в декодированном слове. Поэтому при моделировании для выбора правильного слова из списка использовался критерий корректности CRC, как это было сделано в работе [46]. Для расчета CRC использовался следующий полином [45]

$$g_{\text{CRC24B}}(D) = D^{24} + D^{23} + D^6 + D^5 + D + 1.$$

Т.к. с ростом длины информационного слова эффективность списочного декодирования различна, ниже приведены результаты моделирования для нескольких длин информационных слов.

Поскольку алгоритм Scaled Max-Log-MAP и Log-MAP близки по произво-

дительности, то в качестве алгоритма декодирования компонентных кодов был использован Scaled Max-Log-MAP с коэффициентом 0,75. При генерации списков также использовался алгоритм Scaled Max-Log-MAP.

На рисунках 2.11, 2.13, 2.14 представлены результаты моделирования для различных длин информационного слова для первого подхода. Т.к. моделирование проводилось для канала с АБГШ, то результаты моделирования при генерации списка с помощью второго компонентного кода были такими же, как и для первого, поэтому приведены графики лишь для последнего случая. В качестве критерия сравнения выбрана вероятность ошибки на пакет FER, поскольку основным критерием при работе списочного декодирования является правильность декодированного слова. На рисунке 2.12 представлены результаты сравнения предложенного алгоритма с другими списочными подходами, на которых видно, что предложенный алгоритм показывает лучшие результаты.

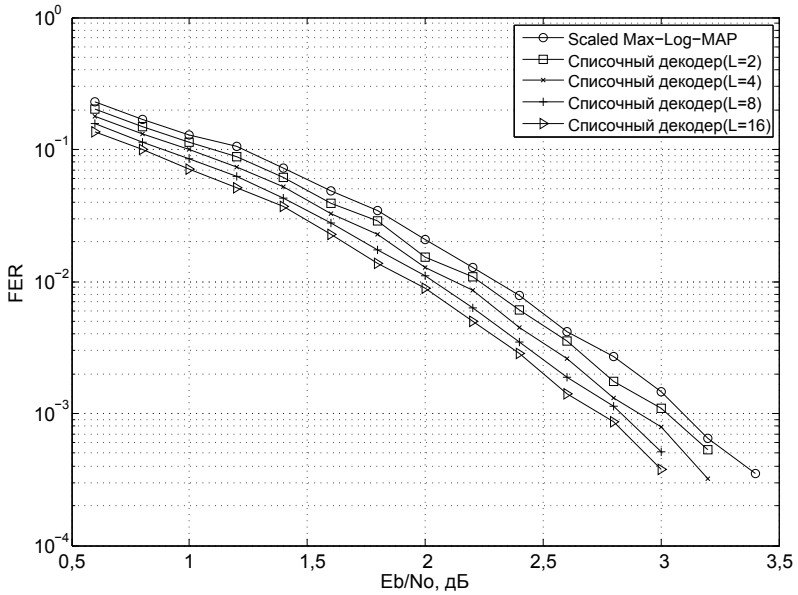

Рисунок 2.11 – Результат моделирования списочного турбо-декодера ($k = 40$)

Из результатов моделирования видно, что на малых длинах информационного слова выигрыш от использования списочного декодера варьируется от 0,1 до 0,4 дБ по вероятности ошибки на пакет для списка от 2 до 16 последовательностей. На больших длинах эффект от списочного декодирования уменьшается, однако в отличие от описанных выше алгоритмов, он достигает не менее 0,1 дБ,

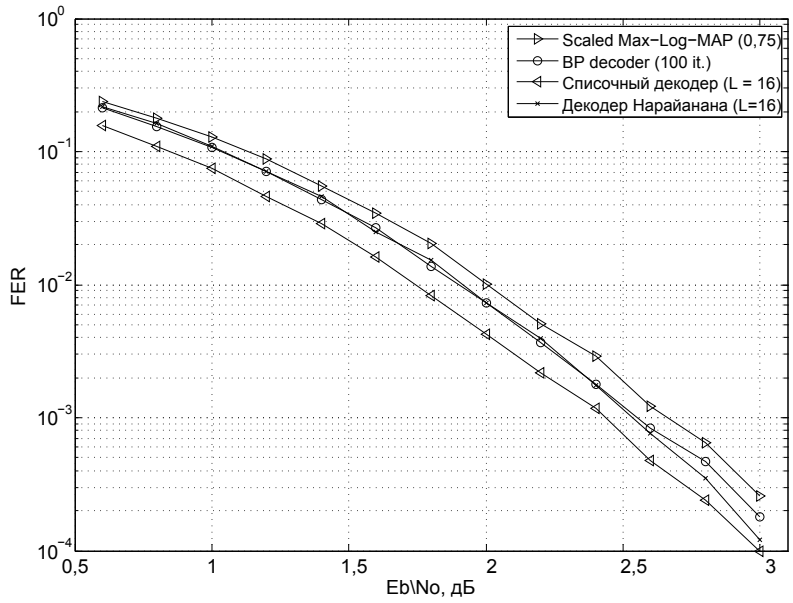

Рисунок 2.12 – Результат моделирования списочного турбо-декодера ($k = 64$)

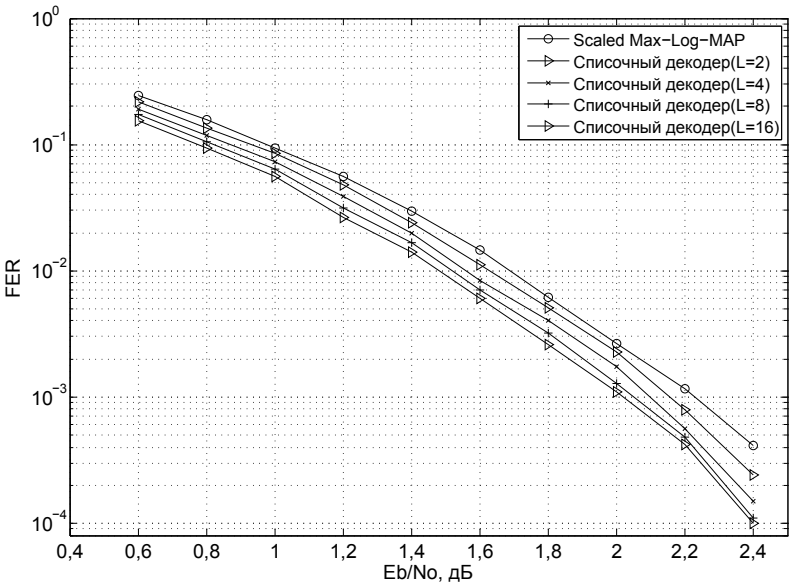

Рисунок 2.13 – Результат моделирования списочного турбо-декодера ($k = 112$)

что не могут обеспечить другие списочные декодеры турбокода с реализуемой сложностью.

На рисунках 2.15, 2.16 показаны результаты моделирования при использовании оконного подхода. Т.к. предложенный списочный декодер показывает лучшие результаты на коротких словах, длина окна была фиксирована в интервале от 40 до 64 бит. При этом решетка была разбита на фрагменты одинаковой длины, размер списка в каждом окне обозначен как L, итоговый размер списка

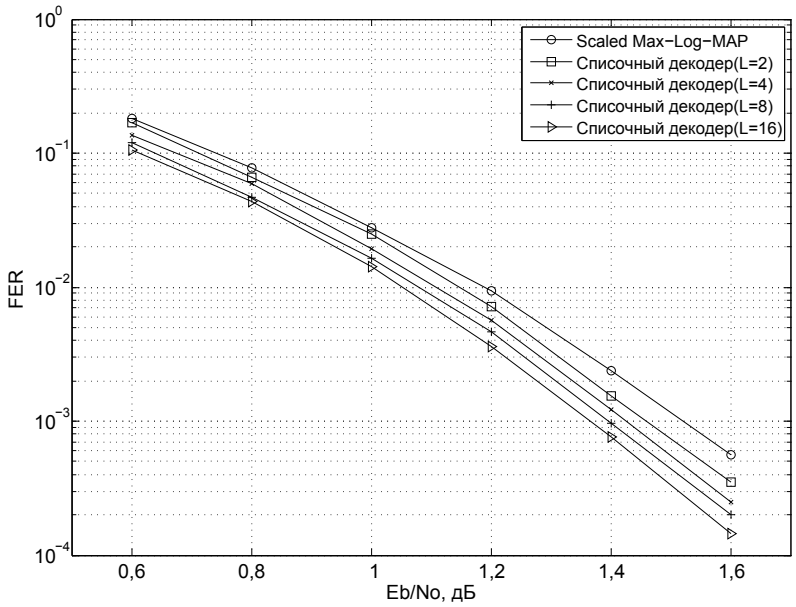

Рисунок 2.14 – Результат моделирования списочного турбо-декодера ($k = 320$)

обозначен как L_f. Для информационного слова длины $k = 112$ бита решетка была разбита на 2 окна, в то время как для $k = 320$ использовалось 8 окон.

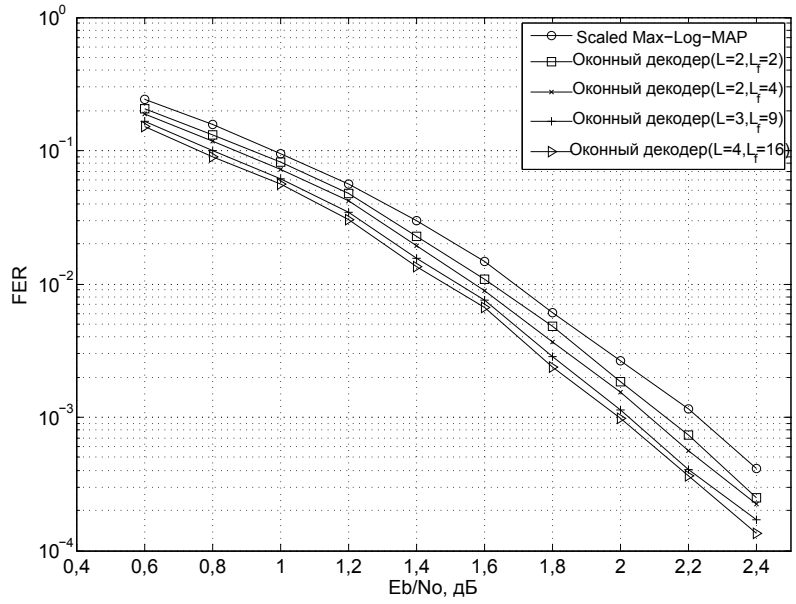

Рисунок 2.15 – Результат моделирования списочного турбо-декодера при оконном подходе к генерации списка($k = 112$)

Как видно из результатов моделирования оконный подход позволяет не только распараллелить вычисление списка, уменьшить время его генерации, но и немного увеличить выигрыш от использования списочного декодера. Это свя-

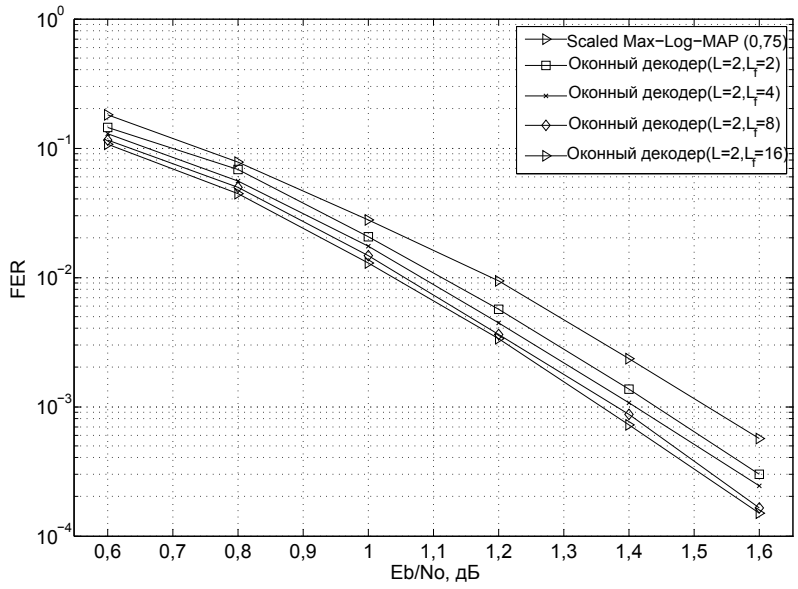

Рисунок 2.16 – Результат моделирования списочного турбо-декодера при оконном подходе к генерации списка($k = 320$)

зано с тем, что в оконном алгоритме достигается большее расхождение между элементами списка, что позволяет достичь лучших результатов на больших длинах информационного слова.

2.5 Заключение и выводы по разделу

В разделе предложен новый алгоритм списочного декодирования турбокода. Были проанализированы существующие алгоритмы списочного декодирования турбокодов, которые основаны на списочном декодере сверточного кода и на алгоритме распространения доверия. Показано, что существующие методы позволяют достичь выигрыш лишь на длинах информационных слов до 40 бит, в то время как на больших длинах выигрыш в области резкого спада вероятности ошибки на пакет, наиболее интересной с практической точки зрения, не виден. Другим недостатком рассмотренных алгоритмов является высокая сложность и задержка.

Разработанный алгоритм лишен этих недостатков. Положительной чертой алгоритма является то, что он позволяет выполнять списочное декодирование в

параллель, что обеспечивает задержку обработки, сравнимую с обычным итеративным декодером. Результаты моделирования в канале с АБГШ показали, что алгоритм позволяет получить выигрыш от 0,15 до 0,4 дБ в зависимости от длины информационного слова для списка длины 16. Также предложена оконная модификация алгоритма генерации списка, которая позволяет ускорить вычисления без ухудшения производительности.

Основные результаты данного раздела можно сформулировать следующим образом:

1. Проанализированы существующие методы списочного декодирования турбокода. Показано, что они не могут быть использованы в практических системах.
2. Предложен новый алгоритм списочного декодирования, позволяющий получить выигрыш по вероятности ошибки на пакет по сравнению с классическим итеративным декодером от 0,15 до 0,4 дБ в зависимости от длины информационного слова в канале с АБГШ.
3. Предложен оконный вариант списочного декодера, который без уменьшения выигрыша на больших длинах информационных слов дает возможность ускорить процесс формирования списка и уменьшить задержку декодирования.

3 Совместное декодирование турбокода и кода источника

3.1 Вводные замечания

В данном разделе рассмотрен другой подход к улучшению производительности систем связи — совместное декодирование кодов источника и канала. Для совместного декодирования источника и канала существует два подхода. Первый связан с построением единой решетки, по которой производится декодирование. Однако этот подход является трудоемким и не реализуем в практических системах. Второй подход основан на использовании турбо-принципа [10], в ходе которого декодеры канала и источника итеративно обмениваются информацией, за счет чего достигается лучшее качество приема. Ниже рассматривается второй метод, для улучшения параметров которого использован принцип параллельного списочного декодирования, предложенный в предыдущем разделе.

Рассмотрим систему передачи данных, изображенную на рисунке 1.5, где как «источник данных» обозначен кодер источника, на который поступают данные определенной природы. Последними могут быть текст, голосовой или видео поток и др. На приемной стороне принятые из канала значения декодирует канальный декодер, после чего информационное слово обрабатывается декодером источника. Такой схемы обычно придерживаются при проектировании и практической реализации передатчика и приемника.

С точки зрения теории кодирования и теории информации задачу сжатия источника и передачи информации по каналу связи можно решать независимо без потери качества для системы передачи в целом [65]. Однако теоремы кодирования, исследующие потенциальные возможности систем связи, доказываются в условиях отсутствия ограничений на длину кодового слова, т. е. в отсутствие ограничений на сложность кодирующих и декодирующих устройств. На практике такие ограничения часто являются принципиальными. Поэтому задачи

совместного кодирования и декодирования канала и источника связи (СККИ и СДКИ) вызывают значительный теоретический и практический интерес.

Несмотря на то, что СККИ исследовано довольно хорошо [66–68], распространения данная техника не получила и в современных системах связи используется раздельное кодирование канала и источника. Также это делает невозможным использование СККИ в ближайшем будущем из-за проблем, связанных с обратной совместимостью с существующими стандартами и сложностью внедрения.

Однако требования к качеству и скорости передачи данных неуклонно растут. Раздельное улучшение и оптимизация декодеров источника и канала исчерпывает себя, и по этой причине в последнее время все большее внимание уделяется задаче СДКИ. В первую очередь это возможно благодаря тому, что в реальных системах при кодировании источника не от всей избыточности удается избавиться, особенно явно это выражено для видео и аудио контента. Во вторых, к пакетам данных прибавляются различные заголовки, которые также обладают избыточностью. В третьих, часто для обнаружения ошибки используют проверки CRC, которые могут быть использованы при декодировании. Таким образом, декодер источника, зная природу избыточности, может попытаться улучшить качество приема, либо может предоставить на декодер канала дополнительную информацию об информационных битах для более корректного их декодирования. Также СДКИ может быть внедрен и использован в современных системах связи без изменения существующих стандартов.

Данный раздел диссертации посвящена задаче совместного декодирования турбокода и источника данных в современных системах передачи данных, как способ улучшения качества приема. Для улучшения параметров СДКИ предложен списочный алгоритм, который использует избыточность источника и итеративную структуру турбо-декодера. Основное внимание уделено совместному декодированию кодера речи и турбокода в системе 3GPP LTE. В разделе 3.2 рассмотрена задача СДКИ применительно к турбокодам. В разделе 3.3 предло-

жен новый алгоритм совместного декодирования турбокода и кодера источника. Раздел 3.4 посвящен описанию передачи речи в системе LTE и речевым кодекам, которые используются в стандарте, также представлены измерения избыточности на выходе речевого кодека. Результаты моделирования предложенного алгоритма применительно к описанной в разделе 3.4 системе даны в разделе 3.5.

3.2 Способы использования избыточности на выходе кодера источника

3.2.1 Описание избыточности на выходе кодера источника

Рассмотрим кодер источника с фиксированной длиной. В дальнейшем не будем делать различий между понятиями источник и кодер источника. Выход такого кодера источника представляет собой последовательность символов одинаковой длины, которые группируются в пакеты (фреймы) на входе кодера канала. В этом случае избыточность кодера источника можно описать как:

1. Различная вероятность появления символов на выходе источника.
2. Межсимвольная зависимость в рамках одного пакета данных.
3. Межсимвольная зависимость между пакетами данных (временная корреляция).

Первый тип избыточности может быть легко использован на канальном декодере, как некоторое априорное знание о вероятностях бит. Применительно к турбокодам это приведет к тому, что на первой итерации декодирования внешние надежности будут равны не нулю, а некоторым значениям, заранее измеренным и известным. Таким образом, встает задача использования второго и третьего типа избыточности. Поскольку их природа одинакова в дальнейшем в целях простоты обозначения и описания все выкладки будут приведены для межсимвольной зависимости в рамках одного пакета, однако они также справедливы и для зависимости символов между пакетами.

Обозначим пакет на выходе кодера источника как $\mathbf{I} = (I_1, I_2, \ldots, I_m)$, где I_i это i-ый символ пакета, длина которого равна q битам. Также обозначим как $\mathbf{u} = (\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_m})$ битовое представление пакета $\mathbf{I}$, где $\mathbf{u_i} = \{u_i^0, \ldots, u_i^{q-1}\}$ битовое представление символа I_i. Каждый символ такого кодера может принимать $Q = 2^q$ значений.

Первый тип избыточности описывается вероятностями $P(I_i)$ и если вероятность того, что некоторый i-ый символ примет одно их Q значений не равна $1/Q$, то данную информацию можно использовать на канальном декодере. Межсимвольная зависимость определяется вероятностями $P(I_i|I_{i-1}, \ldots, I_1)$. Если $P(I_i|I_{i-1}, \ldots, I_1) = 1/Q$, то зависимости нет, в противном случае говорят о наличии последней.

Чаще всего связь между символами рассматривают как Марковский процесс первого порядка, который характеризуется зависимостями между соседними символами, т.е.

$$P(I_i|I_{i-1}, \ldots, I_1) = P(I_i|I_{i-1}). \tag{3.1}$$

Для многих приложений такое предположение справедливо, в частности для речевых кодеков. При этом для источника, который описывается Марковским процессом первого порядка, существует эффективный алгоритм использования избыточности, который будет описан ниже.

3.2.2 Оценка символов Марковского источника первого порядка в канале с АБГШ

Рассмотрим систему, в которой отсутствуют канальный кодер и декодер, а избыточность кодера источника может быть описана как Марковский процесс первого порядка. Также пусть эта зависимость известна как на отправляющей, так и на приемной стороне, как показано на рисунке 3.1. Обозначим последовательность на выходе канала как $\mathbf{r} = (\mathbf{r_1}, \mathbf{r_2}, \ldots, \mathbf{r_m})$, где $\mathbf{r_i} = \{r_i^0, \ldots, r_i^{q-1}\}$ и для которого при условии двоичной фазовой манипуляции в соответствии с

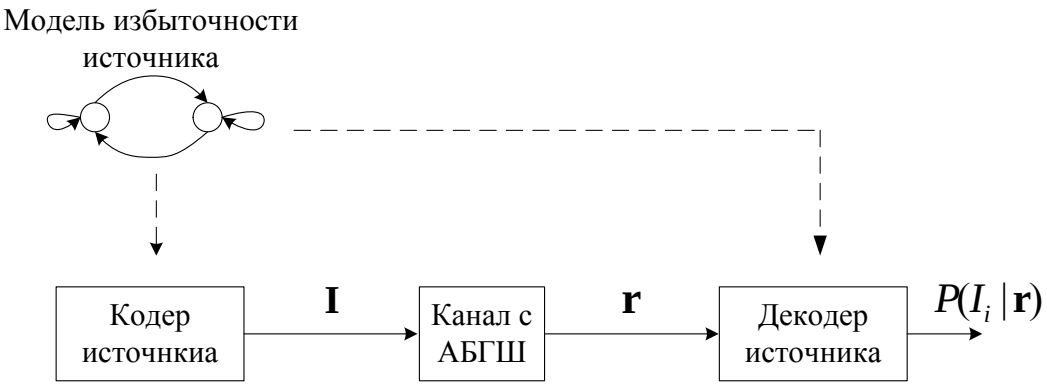

Рисунок 3.1 – Оценка символов источника на приемной стороне

формулой 1.7 справедливо выражение

$$P(r_i^j|u_i^j) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(r_i^j-(2u_i^j-1))^2}. \tag{3.2}$$

Т.к. канала без памяти, то для символа I_i справедливо следующее выражение

$$P(\mathbf{r_i}|I_i) = \prod_{j=0}^{q-1} P(r_i^j|u_i^j). \tag{3.3}$$

Если Марковский процесс, характеризующий источник, описать во времени, то в результате будет получен аналог решетки сверточного кода. Каждый переход в такой решетке соответствует выходу того или иного символа кодера источника. Для рассматриваемого случая решетка будет полносвязной, как показано на рисунке 3.2, где Ψ_i состояние решетки в i-ый момент времени. Поскольку пакет содержит m символов, такая решетка будет содержать m секций, а количество состояний будет зависеть от разрядности символов и будет равно Q.

Для оценки значений символов в такой системе можно использовать алгоритм BCJR [11], который позволяет рассчитать апостериорные вероятности символов $P(I_i|\mathbf{r})$. В результате, оценив вероятности $P(I_i|\mathbf{r})$ для всех $I_i \in \{0, 1, \ldots, Q-1\}$, остается лишь выбирать символ с наибольшей вероятностью в каждой секции решетки.

По аналогии с описанием алгоритма BCJR для сверточных кодов вероятность $P(I_i = p_2|\mathbf{r})$, что символ I_i равен некоторому значению $p_2 \in \{0, 1, \ldots, Q-$

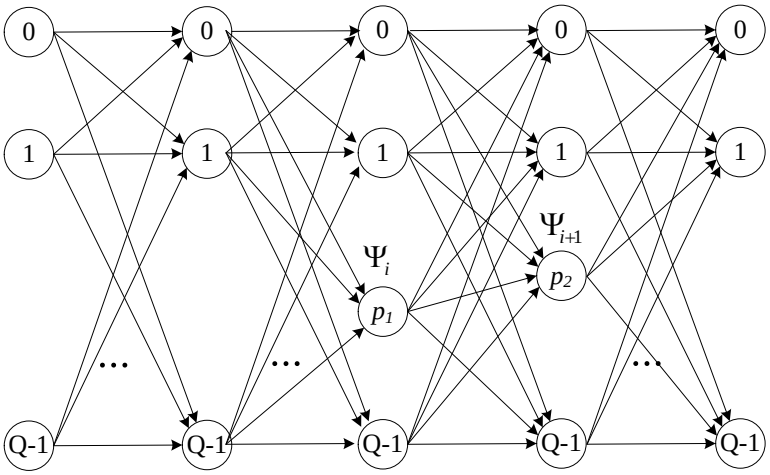

Рисунок 3.2 – Пример полносвязной решетки для Марковского процесса первого порядка

$1\}$, рассчитывается следующим образом

$$\begin{aligned} P(I_i = p_2|\mathbf{r}) &= \sum_{p_1=0}^{Q-1} P(\Psi_i = p_1, \mathbf{r}_{<\mathbf{i}}) P(\mathbf{r}_{>\mathbf{i}}|\Psi_{i+1} = p_2) \times \\ &\times P(\Psi_{i+1} = p_2, \mathbf{r}_\mathbf{i}|\Psi_i = p_1)/P(\mathbf{r}) \\ &= \sum_{p_1=0}^{Q-1} \alpha_i(p_1)\gamma_i(p_1, p_2)\beta_{t+1}(p_2)/P(\mathbf{r}), \end{aligned} \tag{3.4}$$

где значения $\alpha_i(p_1)$ и $\beta_{t+1}(p_2)$ рассчитываются в соответствии с формулами 1.28 и 1.29. Начальные значения $\alpha_0(p_1)$ и $\beta_m(p_2)$ чаще всего считают равновероятными и устанавливают равными $1/Q$.

Основной задачей остается расчет значений $\gamma_i(p_1, p_2)$ для переходов решетки. Используя выражение 1.30 можно получить следующую формулу

$$\gamma_i(p_1, p_2) = P(\mathbf{r}_\mathbf{i}|\Psi_i = p_1, \Psi_{i+1} = p_2) P(\Psi_{i+1} = p_2|\Psi_i = p_1) \tag{3.5}$$

Для канала с АБГШ $P(\mathbf{r}_\mathbf{i}|\Psi_i = p_1, \Psi_{i+1} = p_2)$ рассчитывается в соответствии с выражением 3.3

$$P(\mathbf{r}_\mathbf{i}|\Psi_i = p_1, \Psi_{i+1} = p_2) = \prod_{j=0}^{q-1} P(r_i^j|u_{i,p_2}^j,)$$

где u_{i,p_2}^j это j-ый бит в битовом представление символа p_2. Вероятность $P(\Psi_{i+1} = p_2|\Psi_i = p_1)$ есть ни что иное, как переходная вероятность рассмат-

риваемого Марковского процесса, которая считается известной. Таким образом можно рассчитать вероятности символов в каждой секции решетки. После того, как были рассчитаны символьные вероятности на выходе декодера источника, они могут быть легко преобразованы в битовые $P(u_i^j = x|\mathbf{r})$ с помощью следующего выражения

$$P(u_i^j = x|\mathbf{r}) = \sum_{\text{все} I_i, \text{ где } u_i^j = x} P(I_i|\mathbf{r}), \tag{3.6}$$

где $x \in \{0, 1\}$.

3.2.3 Совместное декодирование кодов канала и источника

В случае если в системе используется кодер канала, то алгоритм BCJR также может быть использован для декодирования источника, однако декодер канального кода должен быть декодером с мягким выходом. Полученные битовые вероятности на выходе канального декодера используются при вычислении вероятности $P(\mathbf{r_i}|I_i)$ в соответствии с формулой 3.3, где вместо вероятностей $P(r_i^j|u_i^j)$ будут стоять значения с выхода декодера канала. Наибольшую популярность описанная техника получила применительно к системам, где используются сверточные коды, таким как GSM и UMTS [69, 70].

Но такая простая коррекция вероятностей на выходе мягкого декодера может быть улучшена, если применить итеративную схему, как это делается в декодере турбокода [69, 71]. Обозначим как $\hat{P}(u_i^j|\mathbf{r})$ рассчитанные вероятности на выходе канального декодера и как $\tilde{P}(u_i^j|\mathbf{r})$ вероятности на выходе декодера источника. Легко заметить, что выходные битовые вероятности алгоритма BCJR канального декодера можно разбить на две составляющие

$$\hat{P}(u_i^j|\mathbf{r}) = \hat{P}_{\text{ch}} \cdot \hat{P}_{\text{ap}}, \tag{3.7}$$

где $\hat{P}_{\text{ap}} = P(u_i^j)$ априорная вероятность бита, а $\hat{P}_{\text{ch}}$ канальная составляющая вероятности, которая зависит от символов принятых из канала. Как и в случае

турбокода в начальный момент времени $\hat{P}_{\text{ap}}$ равны $1/2$, однако в процессе декодирования эти значения могут быть исправлены.

Для декодера источника выходные вероятности бит также можно представить в виде произведения [71]

$$\tilde{P}(u_i^j|\mathbf{r}) = \tilde{P}_{\text{ch}} \cdot \tilde{P}_{\text{ext}}, \tag{3.8}$$

где $\tilde{P}_{\text{ch}}$ равна $\hat{P}_{\text{ch}}$ и поступает от декодера источника, а $\tilde{P}_{\text{ext}}$ составляющая, которая известна благодаря избыточности источника. Как и в случае декодера турбокода, два декодера могут итеративно обмениваться вероятностями для улучшения качества приемника. Пусть выходные значения декодера канала и декодера источника соответственно равны $\hat{P}^{it}(u_i^j|\mathbf{r}) = \hat{P}^{it}_{\text{ch}}\hat{P}^{it}_{\text{ap}}$ и $\tilde{P}^{it}(u_i^j|\mathbf{r}) = \tilde{P}^{it}_{\text{ch}}\tilde{P}^{it}_{\text{ext}}$, где индекс it говорит о номере итерации декодера. Тогда итеративный декодер будет выглядеть как показано на рисунке 3.3. В такой схеме в начале декодирования

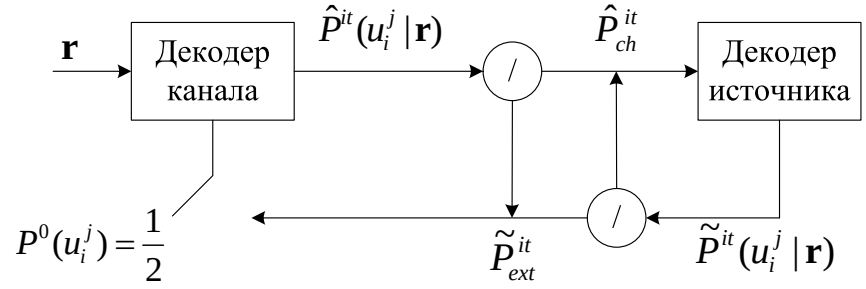

Рисунок 3.3 – Итеративная схема декодирования кодов источника и канала [71]

априорные вероятности на входе сверточного декодера равны $\hat{P}^0_{\text{ap}} = 1/2$, однако на каждой следующей итерации она устанавливается равной $\tilde{P}^{(it-1)}_{\text{ext}}$. На вход декодера источника подаются канальные составляющие $\hat{P}^{it}_{\text{ch}}$. Таким образом, избыточность источника используется при декодировании канального кодера и влияет на все биты в информационном слове, особенно если между кодером канала и кодером источника предусмотрен перемежитель.

3.2.4 Использование совместного декодирования для турбокодов

Если в качестве канального кода используется сверточный код, то схема на рисунке 3.3 может быть легко использована для совместного декодирования. Согласно работам [69, 71, 72] для такого итеративного декодера достаточно 2-3 итераций, после чего рост производительности не наблюдается. Однако в случае турбокода, где сам канальный декодер является итеративным, декодер источника может быть внедрен прямо в итеративный декодер турбокода. В данном случае декодер источника используется для исправления надежностей на выходе каждого декодера компонентного кода, и результирующая схема выглядит как показано на рисунке 3.4.

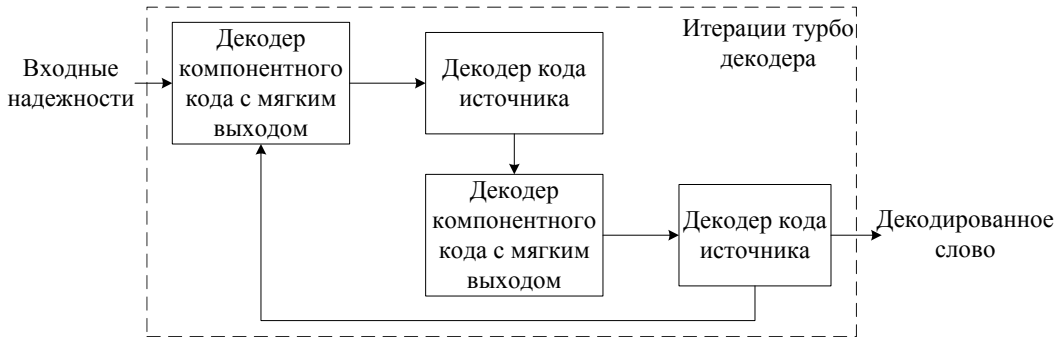

Рисунок 3.4 – Итеративная схема декодирования избыточности источника и турбокода

Часто на практике лишь часть данных на выходе источника обладает избыточностью, к тому же информационное слово содержит протокольные заголовки, которые в данной работе считаются случайными. Тогда при совместном декодировании декодер источника будет использовать известную избыточность лишь части пакета и при итерировании передавать априорные вероятности для соответствующих бит. При таком сценарии применение совместного декодера для сверточного кода даст незначительный выигрыш, поскольку эффект от декодирования источника будет наблюдаться лишь на коррелирующих символах и на битах в некоторой области вокруг рассматриваемого сегмента. Для турбокода избыточность части пакета будет влиять на весь пакет, если используется

представленный на рисунке 3.4 декодер. Это связано с тем, что биты, которые соответствуют коррелированным символам будут перемешиваться перемежителем и во время декодирования второго компонентного декодера влиять на все информационное слово. Пример проиллюстрирован на рисунке 3.5. Как видно на входе первого компонентного декодера данные источника, обладающие избыточностью, окружены некоторыми случайными битами. При совместном декодировании первого компонентного кода турбокода данные источника будут влиять на некоторую область вокруг себя, которая представлена заштрихованной областью. На входе второго компонентного декодера биты информационного слова будут перемешаны, что приведет к тому, что при СДКИ коррелированные биты будут влиять на остальные части слова. Даже если размер коррелированных данных мал, использование совместного турбо-декодирования может дать прирост производительности по вероятности ошибки на пакет, что важно при пакетной передаче данных.

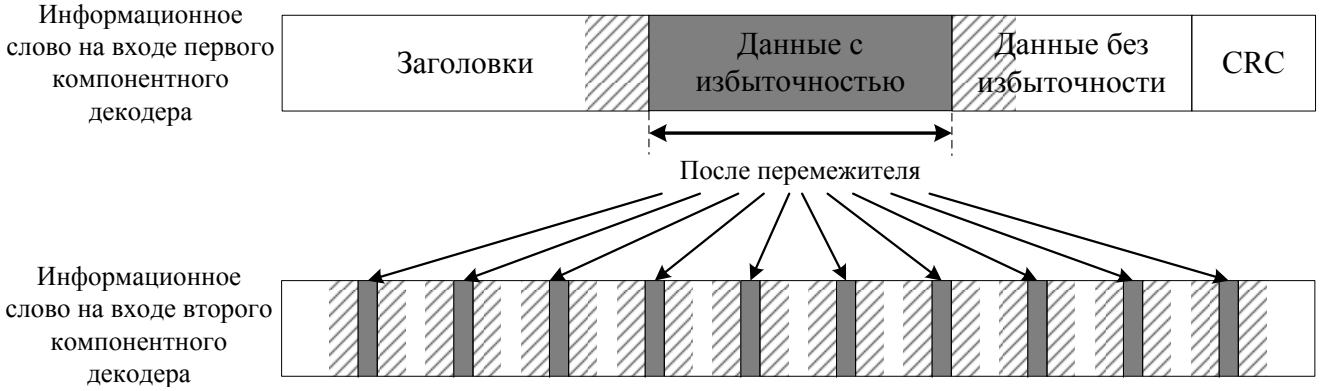

Рисунок 3.5 – Пример влияния коррелированных данных на процесс совместного турбо-декодирования (заштрихованные области соответствуют области влияния СДКИ)

Т.к. информационное слово содержит проверки CRC, то их также можно использовать для улучшения параметров приемника. Проверки CRC могут быть использованы не только для обнаружения ошибок, но и для их исправления [56]. В случае турбокода можно реализовать декодер CRC с мягким выходом по решетке кода и вовлечь его в итеративный процесс. Но улучшение от такого

подхода будет незначительным, поскольку длина защищенного CRC фрагмента может быть большой и вклад от декодирования будет ничтожным. К тому же такой декодер является довольно сложными: для CRC длиной 8 бит необходимо строить решетку с 256 состояниями и большим числом секций. Однако основным фактором против такой стратегии является то, что исчезнет возможность проверки на ошибки декодированного слова, что может нарушить работу вышестоящих уровней и привести к катастрофическим последствиям. Поэтому декодирование CRC в данной работе не рассматривается. Следовательно встает задача максимального использования избыточности на приемнике, которая включала бы в себя декодирование источника, использование CRC и некоторых априори известных значений бит.

3.3 Алгоритм совместного декодирования турбокода и кода источника

Рассмотрим приближенную к реальным условиям ситуацию, когда входом кодера канала является информационное слово, имеющее структуру представленную на рисунке 3.5. Допустим, что проверка CRC вычисляется для всего слова. Такая схема используется во многих системах, к примеру таким образом выглядят информационные слова на входе турбокода в системе 3GPP LTE. Также предположим, что символы на выходе кодера источника (обозначены как «данные с избыточностью») обладают избыточностью, которая может быть описана Марковским процессом первого порядка, в то время как остальные биты будем считать случайными.

Для рассматриваемого случая может быть использован алгоритм СДКИ, который был описан в разделе 3.2.4, однако результат такого подхода будет сильно зависеть от известной избыточности и количества случайных бит. Предположим, что выход источника обладает высокой избыточностью и в результате СДКИ после нескольких итераций совместного турбо-декодера биты источника

были декодированы корректно. Тогда можно говорить о том, что они бесконечно надежны и такие биты можно назвать опорными битами или пилотными. В ходе дальнейшего декодирования они в значительной мере будут влиять на надежности рядом стоящих бит. Если при декодировании турбокода имеется информация о таких опорных битах, то благодаря перемежителю их влияние на декодированное слово будет значительно больше, как было показано на рисунке 3.5. Следовательно, чтобы увеличить шансы корректного декодирования всего слова, надежности успешно декодированных бит источника можно установить равными $\pm\infty$ в зависимости от значения бита. Чтобы распространить влияние высоконадежных позиций на все слово, необходимо выполнить несколько дополнительных итераций классического турбо-декодера, т.к. биты источника уже считаются известными и дальнейшее совместное декодирование не имеет смысла. Если есть возможность проверить корректность бит источника, например если в системе используется отдельное CRC для обнаружения ошибок в данных источника, то такой подход даст улучшение производительности декодера. Если же возможности проверки бит источника на корректность нет можно предположить, что они были декодированы корректно, использовать их как пилотные, а после декодирования выполнить проверку CRC для всего слова на физическом уровне и сделать вывод о корректности предположения.

Однако не всегда благодаря избыточности можно точно декодировать биты источника. Несмотря на это, очевидно, что биты, которые соответствуют коррелированным символам источника, будут декодированы лучше. Т.е. вероятность ошибки на бит в последних будет меньше, чем в других частях декодированного слова. Используя это наблюдение, можно усилить эффект от совместного декодирования. Оставшиеся ошибки в битах кодера источника можно исправить с помощью списочного алгоритма Витерби. Заметим, что на входе первого компонентного кода биты источника локализованы. Если использовать алгоритм Витерби не для всего информационного слова, а лишь для части, где расположены данные с избыточностью, то эффективность списочного декодирования

будет выше. Таким образом, на решетке первого компонентного кода с помощью оконного алгоритма Витерби можно вычислить список слов источника и рассчитывать на то, что будет найдена корректная последовательность.

Собрав все наблюдения и замечания воедино, новый совместный списочный декодер формулируется для двух случаев. Если есть возможность определить правильное слово в списке, то схема результирующего декодера будет выглядеть как показано на рисунке 3.6. В начале выполняется некоторое количество итераций совместного декодера. После этого, если слово не было декодировано корректно, в окне выполняется списочный декодер Витерби и выбирается корректная последовательность. Если такая последовательность была найдена, то биты последовательности устанавливаются как опорные и выполняется несколько итераций обычного турбо-декодера для распространения их влияния на все слово. Информационное слово считается декодированным успешно, если была удовлетворена последняя проверка CRC для всего слова. Если же нет возможно-

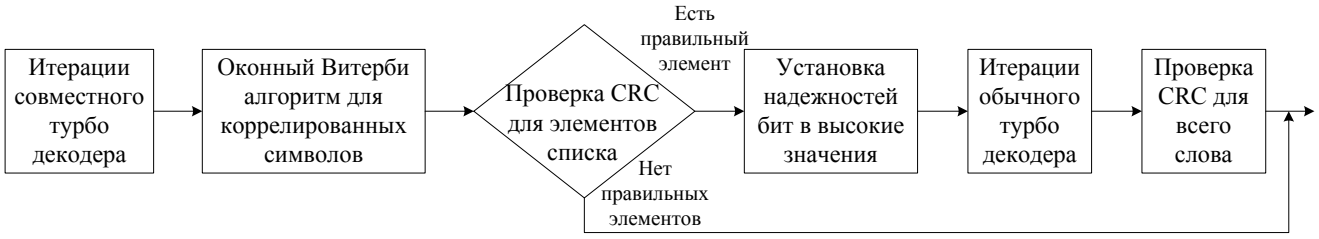

Рисунок 3.6 – Схема совместного турбо-декодера №1

сти проверки корректности последовательности на выходе списочного декодера Витерби, делается предположение, что любой элемент списка может быть потенциально корректным и результирующий декодер примет вид, показанный на рисунке 3.7. Как видно, для каждой последовательности списка значения бит принимаются как надежные и для каждой последовательности выполняются параллельные турбо-декодеры. В результате декодирования будет получен список слов, из которых можно выбрать правильное.

Поскольку производительность СДКИ сильно зависит от источника и других параметров системы, работа предложенного алгоритма и результаты срав-

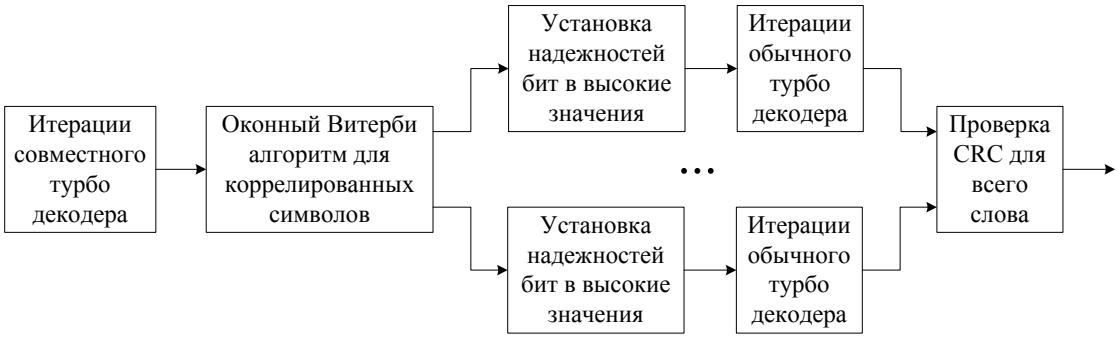

Рисунок 3.7 – Схема совместного турбо-декодера №2

нения с обычным СДКИ декодером приведены на примере передачи голосовых пакетов в система 3GPP LTE.

3.4 Передача речи и избыточность на выходе речевых кодеков в стандарте 3GPP LTE

3.4.1 Кодирование речи в системе 3GPP LTE

В стандарте 3GPP LTE кодирование речи осуществляется с помощью одного из двух алгоритмов: AMR-NB или AMR-WB. Оба кодека поддерживают несколько возможных степеней сжатия или скоростей передачи, которые варьируются от 1,8 до 12,2 Кбит/с для AMR-NB, и от 1,75 до 23,85 Кбит/с для AMR-WB. Режимы работы 12,2 Кбит/с и 12,65 Кбит/с для кодеков AMR-NB и AMR-WB, соответственно, в системе 3GPP LTE считаются основными и для них приведены результаты измерений и моделирования.

AMR-NB является узкополосным кодеком и диапазон частот кодируемой речи ограничен 300 Гц снизу и 3400 Гц сверху [73]. В отличии от AMR-NB AMR-WB является широкополосным и кодирует речь в диапазоне частот от 50 до 7000 Гц и обеспечивает практически идеальное качество голоса на приемной стороне [74]. Помимо механизма кодирования речи в стандарте также описан алгоритм определения активности голоса (VAD) [75, 76]. В случае если алгоритм установил, что в данный момент абонент молчит, то передается информация,

необходимая для синтеза комфортного шума на приемной стороне [77, 78]. В случае потери пакетов AMR определяет механизм восстановления последних. Для этого для синтеза речи используются предыдущие успешно принятые пакеты [79, 80]. Поскольку тема данной диссертационной работы не связана с речевыми кодеками, далее будет дано краткое описание принципов кодирования речи в AMR и параметров кодека, необходимое для понимания результатов моделирования. Более детальное и глубокое описание механизма VAD и алгоритма коррекции потерянных пакетов может быть найдено в перечисленных выше стандартах.

Сжатие в AMR происходит по принципу кодирования параметров модели речевого тракта человека. Эта модель может быть представлена как некоторое возбуждение, которое подается на вход акустического артикуляторного фильтра [81], параметры которых меняются во времени. На коротких интервалах времени (5-30 мс) голосовой сигнал можно считать стационарным и, следовательно, можно вычислить и передать параметры модели, с помощью которых речь синтезируется на приемной стороне.

В общем виде источник возбуждения имеет две составляющие, шумовую и периодическую. Периодическая составляющая обусловлена работой голосовых связок и преобладает для гласных и звонких звуков, в то время как шумовая составляющая является доминирующей для шипящих звуков [82]. Основными параметрами для синтеза речи на приемной стороне являются коэффициенты артикуляторного фильтра, период основного тона, характеризующий периодическую составляющую возбуждения, значения шумовой составляющей, коэффициенты усиления обеих составляющих возбуждения. Следует сказать, что значение периода основного тона для одного человека обычно сильно не меняется во времени, в то время как шумовая составляющая имеет случайный характер и практически не обладает корреляцией, которую можно было бы использовать при декодировании. Также стоит отметить, что при синтезе речи наиболее важными являются параметры, относящиеся к фильтру и периодической составляющей

возбуждения.

Расчет параметров в AMR происходит на основе метода линейного предсказания (Linear Predictive Coding, LPC), который хорошо согласуется с моделью речеобразования [81], рассмотренной выше. На вход кодек принимает блоки отсчетов (фреймы), которые соответсвуют 20 мс речевого сигнала, и для каждого блока выполняется анализ. Т.к. входной сигнал для AMR-NB дискретизирован с частотой 8 кГц, а для AMR-WB с частотой 12,8 кГц, то фреймы двух кодеков содержат 160 или 256 отсчетов, соответственно. В основе кодека AMR лежит алгоритм ACELP, схема декодера которого представлена на рисунке 3.8. Как видно в алгоритме ACELP возбуждение представляет собой сумму последовательностей адаптивной и фиксированной кодовой книги. Выход адаптивной кодовой книги зависит от периода основного тона человека, в то время как последовательность в фиксированной книге выбирается так, чтобы минимизировать ошибку между синтезированной речью и исходным сигналом. Параметры возбуждения подбираются на основе принципа «анализ-через-синтез», в котором расчет значений параметров происходит в цикле. На каждой итерации алгоритм с помощью подобранных параметром синтезирует сегмент речи и сравнивает с исходной, и выбирает параметры так, чтобы синтезированная речь была максимально похожа на исходную. Для расчета параметров возбуждения речевой фрейм делится на 4 под-фрейма по 5 мс каждый, и для каждого из них выполняется описанный выше анализ. Параметры артикуляторного фильтра высчитываются для всего 20 мс фрейма.

В AMR-NB артикуляторный фильтр описывается как фильтр 10-ого порядка, коэффициенты которого преобразуются в полюса передаточной функции речевого тракта в частотной области, упорядоченные по возрастанию частоты LSF (Linear Spectral Frequency) [84]. Для дальнейшей передачи они группируются и совместно квантуются с помощью векторного квантования, метод группировки зависит от режима работы кодера. Поскольку LSF значения между фреймами обладают высокой корреляцией [85], перед квантованием из полученных

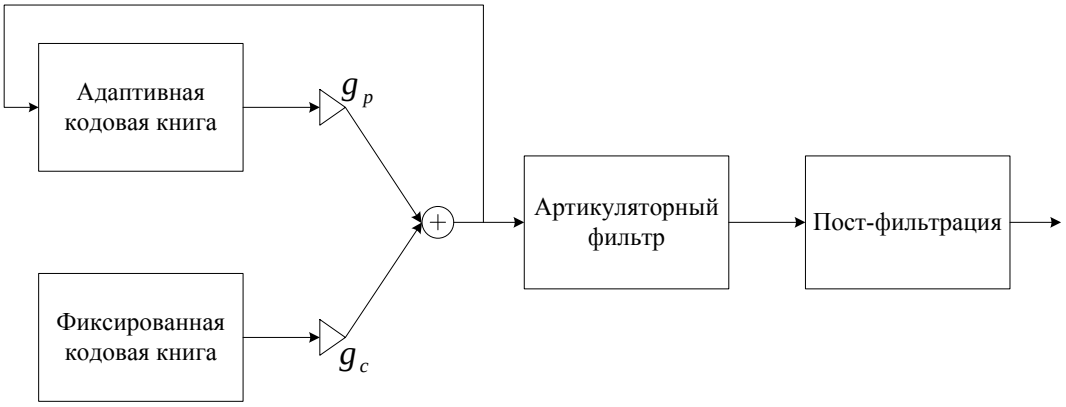

Рисунок 3.8 – Схема декодера ACELP [83]

значений вычитаются вычисленные средние значения LSF коэффициентов. В режиме передачи 12,2 Кбит/с высчитываются параметры для двух фильтров 10-ого порядка, которые также переводятся в LSF коэффициенты, группируются и квантуются с помощью матричного квантования, в результате чего получается 5 индексов, которые необходимо передать. Значение периода основного тона, который в AMR называется как индекс адаптивной кодовой книги, передается лишь для первого и третьего под-фреймов, в то время как для второго и четвертого передается лишь значение приращения относительного значения в предыдущем под-фрейме. Также для каждого под-фрейма вычисляются кодовые последовательности фиксированной кодовой книги, и коэффициенты усиления адаптивной и фиксированной кодовых книг. Как и в случае LSF коэффициентов, перед квантованием из значения коэффициента усиления фиксированной кодовой книги вычитается вычисленное среднее значение. Чем меньше скорость передачи в кодеке AMR-NB, тем большему квантованию подвергаются параметры, что ведет к меньшей остаточной избыточности, т.е. в режиме 12,2 Кбит/с параметры обладают наибольшей корреляцией. Пример формирования AMR фрейма для режима 12,2 Кбит/с и количество бит используемых для передачи представлено в таблице 3.1.

В AMR-WB для лучшего качества синтезированного голоса используется артикуляторный фильтр 16-ого порядка для всех режимов, коэффициенты которого преобразуются в ISP (Immittance Spectral Pairs) [86] значения и сов-

Таблица 3.1 – Распределение бит фрейма AMR-NB в режиме 12,2 Кбит/с [83].

Параметры	Под-фреймы				Всего на фрейм
	1	2	3	4	
2 LSP последовательности					38
Индекс адаптивной кодовой книги	9	6	9	6	30
Коэффициент усиления (g_p)	4	4	4	4	16
Алгебраический код	35	35	35	35	140
Коэффициент усиления (g_c)	5	5	5	5	20
Всего					244

местно квантуются с помощью векторного квантования. Для рассматриваемого режима сжатия в результате квантования получается 7 коэффициентов, которые имеют различный диапазон возможных значений. Период основного тона рассчитывается и передается также, как это делается для AMR-NB. Коэффициенты усиления для обеих кодовых книг рассчитываются для каждого под-фрейма и квантуются совместно. В AMR-WB при различных режимах работы отличаются количество бит, выделенных для передачи кодовых последовательностей фиксированной кодовой книги, от чего также зависит качество синтезируемой речи. При этом количество бит, выделенных на передачу основных параметров, не меняется, что говорит о том, что AMR фрейм в режиме 12,65 Кбит/с обладает наибольшей избыточностью. Пример формирования AMR фрейма для режима 12,65 Кбит/с дан в таблице 3.2.

3.4.2 Обработка AMR пакетов в системе 3GPP LTE

Современные системы связи, такие как Wi-Max и LTE, полностью перешли на передачу данных с коммутацией пакетов, что позволяет сделать архитектуру сети более плоской и прозрачной. Речевые пакеты в системе LTE, как и другие данные, представляют собой IP пакеты, которые обрабатываются нижними

Таблица 3.2 – Распределение бит фрейма AMR-WB в режиме 12,65 Кбит/с [87].

Параметры	Под-фреймы				Всего на фрейм
	1	2	3	4	
VAD-flag					1
ISP коэффициенты					46
Флаг LTP фильтрации	1	1	1	1	4
Индекс адаптивной кодовой книги	9	6	9	6	30
Алгебраический код	36	36	36	36	144
Коэффициенты усиления	7	7	7	7	28
Всего					253

уровнями и передаются абоненту либо от него.

При формировании AMR пакета происходит разделение и группирование бит, соответствующих различным параметрам кодека, на три класса: A, B и C [88, 89]. Биты класса A являются самыми чувствительными к ошибкам и главным образом содержат данные, относящиеся к артикуляторному фильтру и адаптивной кодовой книге. Биты класса B и C менее чувствительны к ошибкам, и содержать данные фиксированной кодовой книги.

Дальнейшая обработка речевого пакета происходит следующим образом:

1. К речевому пакету на выходе кодека добавляются заголовок RTP пакета [90] и RTP заголовок [91], который используется для передачи данных реального времени. Т.к. голосовые данные являются чувствительными к задержкам, то передача сформированного RTP пакета осуществляется протоколами без гарантии доставки UDP/IP [92].
2. Далее пакет поступает на PDCP уровень. На данном уровне для пользовательских данных производится сжатие заголовков алгоритмом ROHC (Robust Header Compression), шифрование данных, контроль очередности и повторения пакетов, приходящих с нижних уровней, управление данными в ходе перехода абонента из одной сети в другую. Сжатие заголовков

верхних уровней является обязательным в случае, когда передаются речевые данные. Если используется протокол IPv4 или IPv6, то заголовки RTP/UDP/IP занимают порядка 40 или 60 байт, соответственно, т.е. практически 50% пакета. Но поскольку RTP/UDP/IP заголовки обладают большой избыточностью, то с помощью алгоритма ROHC [93] их удается сжать до 2-3 байт. Для обеспечения целостности и конфиденциальности передаваемых данных поддерживаются алгоритмы SNOW 3G и AES-128, которые работают в режиме гаммирования. Шифрация является не обязательной для пользовательских данных и в исследованиях предполагалось, что она отключена.

3. Если это необходимо, происходит сегментация или объединение PDCP пакетов, за которую ответственен RLC уровень. RLC уровне отвечает за контроль очередности принятых пакетов и операцию перепосылки данных ARQ (Automatic Repeat Request). RLC уровень может работать в трех режимах: без обработки пакета (TM), с выполнением операции ARQ для данных (AM) и без последней (UM). Режим TM предусмотрен для широковещательной информации. Данные, которые чувствительны к ошибками, но для которых не важна задержка обрабатываются в режиме AM. UM режим использует чувствительный к задержкам трафик. Поскольку речевые пакеты чувствительны к задержкам, то для них является обязательным использование режима UM.
4. RLC пакет обрабатывается MAC уровнем, которые отвечает за выделение физических ресурсов для передачи пользовательских данных, выбор скорости кодирования и вида модуляции, а также операцию гибридной перепосылки данных HARQ (Hybrid Automatic Repeat Request). Для надежной доставки голосовых данных помимо обычного режима поддерживается специальный режим HARQ, когда один голосовой пакет отсылается в четырех последовательных окнах передачи без ожидания прихода квитанции об успешности приема. Такой подход позволяет улучшить качество

приема на границах соты в случае плохого канала.

5. После обработки на MAC уровне данные подаются на физический уровень, где они подготавливаются для транспортировки по беспроводному каналу. Для контроля ошибок пакет защищается 24 битным CRC, после чего производится турбо-кодирование, контроль скорости кода, скремблирование и модуляция. За счет использования турбокодов, технологий MIMO и модуляции OFDM удается достичь высоких скоростей передачи данных.

Заголовок RTP пакета может быть сгенерирован в двух режимах: с минимальной длиной или с выравниванием по длине байта [90]. С точки зрения беспроводной связи больший интерес представляет режим передачи с минимальной длиной. Однако в случае, когда используется режим выравнивания по длине, стандарт прописывает возможность использовать 8-ми битный CRC код для обнаружения ошибок в битах класса A AMR пакета. Т.к. биты класса A содержат наиболее важные данные для восстановления голоса, такая проверка может быть использована для более качественного приема. Стандарт обязывает устройства поддерживать оба варианта.

На уровнях PDCP/RLC/MAC к поступающим данным добавляются свои заголовки, длина которых варьируется от 1 до 3 байт. В данной работе заголовки считаются случайными и не известными, но необходимо отметить, что некоторые поля содержат параметры, которые не изменяются после установления сеанса связи. Другие поля являются счетчиками для отслеживания очередности принятых пакетов, которые также можно учитывать. Такие биты можно считать известными и использовать на практике как пилотные для улучшения параметров декодирования.

3.4.3 Избыточность на выходе речевых кодеков AMR-NB и AMR-WB

Для оценки избыточности на выходе речевых кодеков проанализированы мужская и женская русская речь длительностью от 1,5 до 2 часов. Оба сигнала

подверглись сжатию с помощью одного из алгоритмов AMR-NB или AMR-WB и по полученному потоку была собрана статистика по различным параметрам, которые в дальнейшем рассматривалась независимо. Т.е. считалось, что параметры различной природы никак не связаны между собой.

Для оценки корреляции в рамках одного фрейма с помощью собранной статистики оценены вероятности $P(\Psi_i^t)$ и $P(\Psi_{i-1}^t, \Psi_i^t)$, где $\Psi_i^t \in \{0, \ldots, Q-1\}$ значение i-ого параметра в t-ом фрейме. Условная вероятность $P(\Psi_i^t|\Psi_{i-1}^t)$, характеризующая внутри-фреймовую корреляцию, рассчитывалась по следующему выражению

$$P(\Psi_i^t|\Psi_{i-1}^t) = \frac{P(\Psi_{i-1}^t, \Psi_i^t)}{P(\Psi_{i-1}^t)}.$$

Таким же образом производился расчет вероятностей $P(\Psi_i^t|\Psi_i^{t-1})$, описывающих межсимвольную корреляцию между фреймами:

$$P(\Psi_i^t|\Psi_i^{t-1}) = \frac{P(\Psi_i^{t-1}, \Psi_i^t)}{P(\Psi_i^{t-1})},$$

где $P(\Psi_i^{t-1}, \Psi_i^t)$ были получены из собранной статистики.

С помощью полученных вероятностей для различных параметров были рассчитаны энтропия

$$H(\Psi) = -\sum_{k=0}^{Q-1} P(\Psi = k) \log_2 P(\Psi = k) \tag{3.9}$$

и условная энтропия для коррелированных параметров

$$H(\Psi_i^t|\Psi_{i-1}^t) = -\sum_{k=0}^{Q-1}\sum_{j=0}^{Q-1} P(\Psi_i^t = k, \Psi_{i-1}^t = j) \log_2 P(\Psi_i^t = k|\Psi_{i-1}^t = j), \tag{3.10}$$

$$H(\Psi_i^t|\Psi_i^{t-1}) = -\sum_{k=0}^{Q-1}\sum_{j=0}^{Q-1} P(\Psi_i^t = k, \Psi_i^{t-1} = j) \log_2 P(\Psi_i^t = k|\Psi_i^{t-1} = j). \tag{3.11}$$

Избыточность параметра измерялась как разность между количеством бит, которые используются для передачи по стандарту и измеренным значением энтропии [94]:

$$\Delta R(H(\Psi)) = q - H(\Psi)$$

и

$$\Delta R(H(\Psi_i^t|\Psi_{i-1}^t)) = q - H(\Psi_i^t|\Psi_{i-1}^t),$$

$$\Delta R(H(\Psi_i^t|\Psi_i^{t-1})) = q - H(\Psi_i^t|\Psi_i^{t-1}).$$

В результате анализа получено, что наибольшей корреляцией обладает индекс адаптивной кодовой книги, поскольку период основного тона, который он характеризует, мало меняется во времени. Также большой межфреймовой зависимостью обладают первый и второй индексы, отвечающие за передачу информации о коэффициентах артикуляторного фильтра. Меньшей избыточностью обладают коэффициенты усиления обеих кодовых книг. Во всех остальных параметрах избыточность практически отсутствует, в то время как кодовые последовательности фиксированной кодовой книги имеют случайный характер. Также из полученных результатов сделан вывод, что биты, которые соответствуют наиболее коррелированным параметрам, относятся к классу A AMR пакета. Поскольку для женского и мужского исследуемых голосов получены практически одинаковые результаты, то приведены замеры лишь для мужской речи. Результаты измерений даны для наиболее коррелированных параметров. Для параметров AMR-NB для межфреймовой корреляции и для внутри-фреймовой корреляции результаты приведены в таблицах 3.3 и 3.4. Для параметров AMR-WB результаты анализа приведены в таблицах 3.5 и 3.6.

3.4.4 Совместный декодера вокодеров семейства AMR и турбокода

Как видно из полученных результатов, лишь часть AMR потока содержит избыточность, которую можно использовать при совместном декодировании канала и источника. Однако благодаря тому, что при формировании AMR пакета биты данных переставляются, биты наиболее коррелированных параметров собираются в классе A. В результате на входе кодера турбокода пакет AMR выглядит как показано на рисунке 3.9 и для его декодирования может быть исполь-

Таблица 3.3 – Результаты анализа избыточности мужской русской речи (AMR-NB, корреляция параметров внутри фрейма)

Параметр	$\Delta R(H(\Psi))$	$\Delta R(H(\Psi_i^t \vert \Psi_{i-1}^t))$
Коэффициент усиления адаптивной кодовой книги	0,1	0,46
Индекс адаптивной кодовой книги	0,17	2,82
Индекс адаптивной кодовой книги (величина приращения)	0,57	0,75
Коэффициент усиления фиксированной кодовой книги	0,77	1,22

зован предложенный списочный совместный декодер. На рисунке рассмотрен случай, когда заголовок RTP пакета содержит CRC и как видно окно списочного Витерби алгоритма включает проверку CRC и биты класса A. Если заголовок не содержит проверки CRC, то окном будет область, где расположены биты класса A.

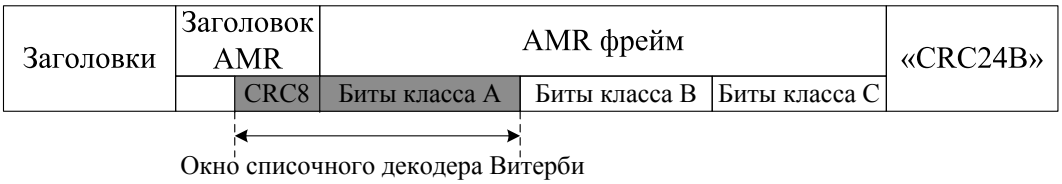

Рисунок 3.9 – Схема AMR пакета

При выполнении совместного декодирования предполагалось, что избыточность всех перечисленных параметров используется независимо. Т.е. декодер источника может быть представлен как группа отдельных декодеров для индексов фильтра, коэффициентов усиления адаптивной и фиксированной кодовых книг, индекса адаптивной кодовой книги.

Таблица 3.4 – Результаты анализа избыточности мужской русской речи (AMR-NB, корреляция параметров между фреймами)

Параметр	$\Delta R(H(\Psi))$	$\Delta R(H(\Psi_i^t \mid \Psi_i^{t-1}))$
LSF коэффициент:		
первый	1,43	2,16
второй	0,88	1,86
Индекс адаптивной кодовой книги:		
первый	0,17	2,46
второй	0,17	2,47
Коэффициент усиления фиксированной кодовой книги:		
первый	0,74	1,21
второй	0,79	1,28
третий	0,79	1,26
четвертый	0,76	1,24

Таблица 3.5 – Результаты анализа избыточности мужской русской речи (AMR-WB, корреляция параметров внутри фрейма)

Параметр	$\Delta R(H(\Psi))$	$\Delta R(H(\Psi_i^t \mid \Psi_{i-1}^t))$
Коэффициент усиления (совместное значение)	0,40	1,35
Индекс адаптивной кодовой книги	0,20	2,26
Индекс адаптивной кодовой книги (величина приращения)	0,39	0,55

Таблица 3.6 – Результаты анализа избыточности мужской русской речи (AMR-WB, корреляция параметров между фреймами)

Параметр	$\Delta R(H(\Psi))$	$\Delta R(H(\Psi_i^t \mid \Psi_i^{t-1}))$
ISP коэффициент:		
первый	0,71	2,39
второй	0,51	1,57
Коэффициент усиления (совместное значение):		
первый	0,39	1,17
второй	0,39	1,18
третий	0,40	1,24
четвертый	0,44	1,29
Индекс адаптивной кодовой книги:		
первый	0,20	1,87
второй	0,21	1,88

3.5 Результаты моделирования в канале с АБГШ

Используя полученные результаты анализа избыточности кодеков AMR-NB и AMR-WB, проведено моделирования для сравнения предложенного списочного алгоритма и обычного итеративного декодера турбокода. Схема системы, на которой проводилось моделирование показана на рисунке 3.10. Размеры заголовков PDCP/RLC/MAC уровней были приняты равными 8 битам каждый, размер сжатых RTP/UDP/IP данных бил принят равным 16 битам. Также при моделировании был отдельно рассмотрен случай, когда заголовок RTP пакета содержит проверку CRC для класса A бит, и когда такой проверки нет.

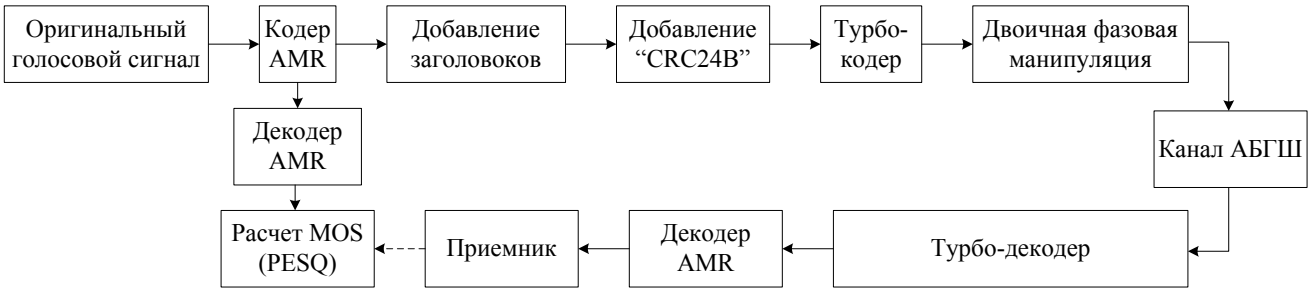

Рисунок 3.10 – Схема модели для тестирования совместного декодера турбокода и вокодера AMR

Таким образом, с учетом заголовков и 24 битного CRC физического уровня, длина пакета для AMR-NB в режиме 12,2 Кбит/с была принята равной 320 бит. Т.к. длина заголовков не зависит от типа кодека, то размер AMR-WB пакета с учетом требований стандарта 3GPP LTE [95] был установлен равным 352 битам. При декодировании выполнялось 8 итераций турбо-декодера, где в качестве декодера компонентного кода использовался алгоритм Scaled Max-Log-MAP со взвешивающим коэффициентом 0,75. Для нового предложенного декодера после 8 итераций совместного декодирования и построения списка выполнялось 8 итераций обычного турбо-декодера. Длина списка алгоритма Витерби была принята равной 16.

При моделировании предполагалось, что передаются только голосовые

AMR фреймы, избыточность которых известна, в то время как пакеты с информацией о комфортном шуме, всегда считались принятыми успешно. Такие пакеты являются короткими и имеют размер всего 39 или 40 бит [88, 89] в зависимости от типа вокодера. Допущение было сделано из предположения, что при передаче таких фреймов в реальной системе есть возможность передать больше избыточных бит и, тем самым, надежно принять короткие пакеты.

Поскольку в стандарте 3GPP LTE на физическом уровне принятые с ошибкой пакеты отбрасываются, то для сравнения алгоритмов была использована вероятность ошибки на пакет. Основное внимание уделено вероятностям ошибки на пакет до 10^{-3}, наиболее интересным с практической точки зрения. Поскольку FER не отображает улучшения качества голоса на приемной стороне, вызванное использованием улучшенного декодера, дополнительно выполнена оценка улучшения качества голоса на приемной стороне. Для сравнения качества принятого голоса был использована метрика MOS (Mean Opinion Score), которая принимает принимает значения от 1 до 5 [96]. Расшифровка значений шкалы MOS приведены в таблице 3.7.

Таблица 3.7 – Шкала качества речи (MOS).

Оценка MOS	Качество речи
5	Прекрасное
4	Хорошее
3	Удовлетворительное
2	Низкое
1	Плохое

В общем виде оценка качества голоса производится группой людей в лабораторных условиях, где выдержаны определенные требования к используемой аппаратуре и помещению, в которых происходит прослушивание [96]. Каждый слушатель оценивает качество исследуемого сигнала по 5 бальной шкале MOS,

после чего рассчитывается среднее значение, которое является искомой оценкой. Однако такой способ дорог и требует много времени. Поэтому институтом ITU-T был стандартизован алгоритм PESQ [97, 98], который в настоящее время является общепринятым алгоритмом оценки качества речи. Алгоритм принимает на вход переданный и принятый речевые сигналы и рассчитывает качество полученного сигнала. При расчете оценки учитывается специфика человеческого речеобразования и восприятия. Строго говоря, рассчитанные алгоритмом PESQ значения не являются значениями MOS, однако могут быть переведены в шкалу MOS по специальному правилу [99]. Полученная величина обозначается как MOS-LQO (Listening Quality Objective). При преобразовании метрики PESQ в MOS-LQO по стандарту [99] предполагается, что максимально возможное значение MOS-LQO равно 4,5 для узкополосных сигналов и 4,64 для широкополосных.

В результате моделирования получено, что предложенный алгоритм для кодека AMR-NB позволяет получить выигрыш чуть больше 0,15 дБ по вероятности ошибки на пакет по сравнению с обычном турбо-декодером, что видно на рисунках 3.11, 3.12, и этот результат не зависит от того учитывается CRC или нет. Это соответствует выигрышу от 0,15 до 0,4 пунктов по MOS (рисунки 3.15, 3.16). Как видно на рисунках 3.13, 3.14, 3.17, 3.18 для AMR-WB выигрыш по FER составляет чуть меньше 0,15 дБ, однако выигрыш по MOS варьируется от 0,18 до 0,4 пунктов. Меньший выигрыш по FER для AMR-WB по сравнению с AMR-NB связан с тем, что отношение числа коррелированных и случайных бит у AMR-NB больше. Также полученные результаты говорят о том, что AMR-WB является более чувствительным к ошибкам, что видно при сравнении метрики MOS.

Из результатов моделирования видно, что если корректность бит источника можно оценить по CRC, то при небольшом усложнении декодера можно получить значительный выигрыш по качеству принимаемой речи. Если такой возможности нет, то используя критерий остановки итеративного декодирования

можно уменьшить число итераций декодера после построения списка и снизить время декодирования.

3.6 Заключение и выводы по разделу

В данном разделе рассмотрена задача совместного декодирования турбокода и кодера источника для улучшения качества приема данных. С учетом специфики формирования данных в современных системах связи предложен новый совместный списочный декодер турбокода, который может быть применен для различных типов источников и стандартов передачи.

Для иллюстрации результатов работы представленного алгоритма рассмотрена задача передачи голоса в стандарте 3GPP LTE, где в качестве источника данных рассмотрены вокодеры AMR-NB и AMR-WB. Проведенный анализ для основных режимов кодирования речи показал, что выходной поток данных кодеков обладает избыточностью, которую можно использовать для улучшения качества принимаемого голоса в системе 3GPP LTE. Результаты анализа избыточности вокодеров приведены в данном разделе диссертационной работы.

Для кодеков AMR-NB и AMR-WB и турбокода предложен совместный декодер, для которого проведено моделирование в канале с АБГШ. Для сравнения алгоритмов использовалась не только метрика FER, но и качество MOS принимаемой речи. Для обоих кодеков предложенный алгоритм показал прирост производительности по сравнению с обычным турбо-декодером. Выигрыш по FER для AMR-NB и AMR-WB составил порядка 0,15 дБ, в то время как выигрыш по MOS варьировался от 0,15 до 0,4 пунктов в зависимости от E_b/N_0.

Таким образом, можно сформулировать следующие основные результаты данного раздела:

1. Предложен совместный списочный декодер турбокода и кода источника для современных мобильных систем связи, при проектировании которого учтено, что лишь часть бит информационного слова обладает известной

избыточностью.

2. Проанализирована избыточность на выходе речевых кодеков AMR-NB и AMR-WB и показано, что сжатый битовый поток обладает корреляцией, которая может быть использована при совместном декодировании.
3. Предложен совместный декодер турбокода и вокодеров семейства AMR для систем связи стандарта 3GPP LTE, который позволяет получить выигрыш по качеству принимаемой речи 0,15 до 0,4 пунктов MOS.

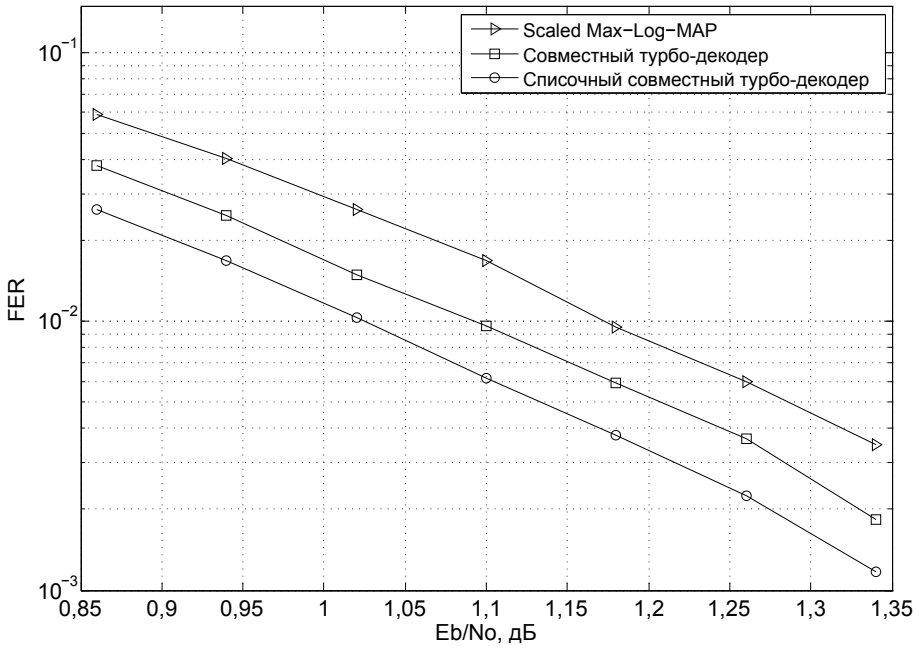

Рисунок 3.11 – Результаты моделирования совместного декодера для вокодера AMR-NB с учетом проверки CRC (FER)

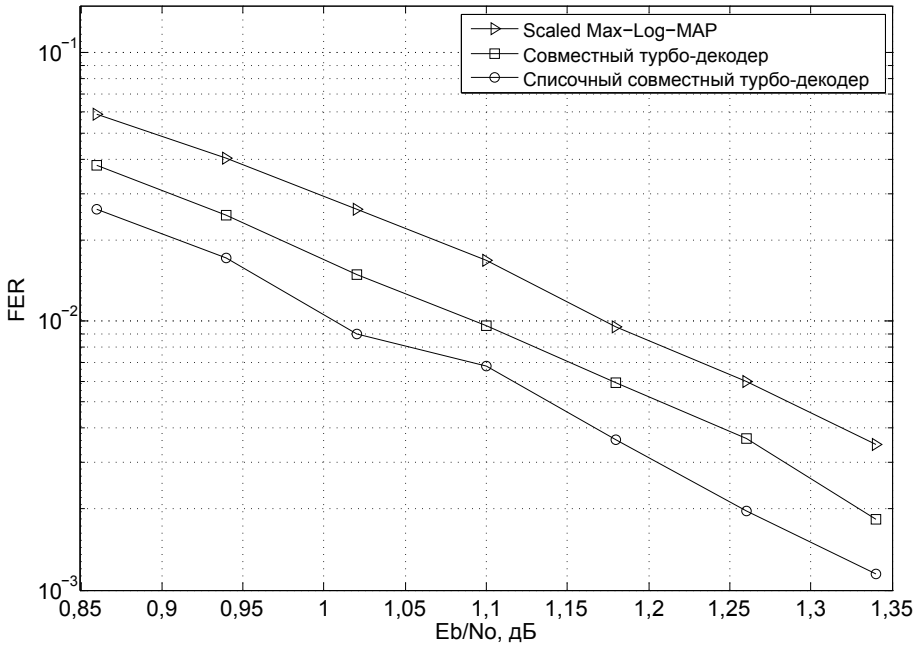

Рисунок 3.12 – Результаты моделирования совместного декодера для вокодера AMR-NB без учета проверки CRC (FER)

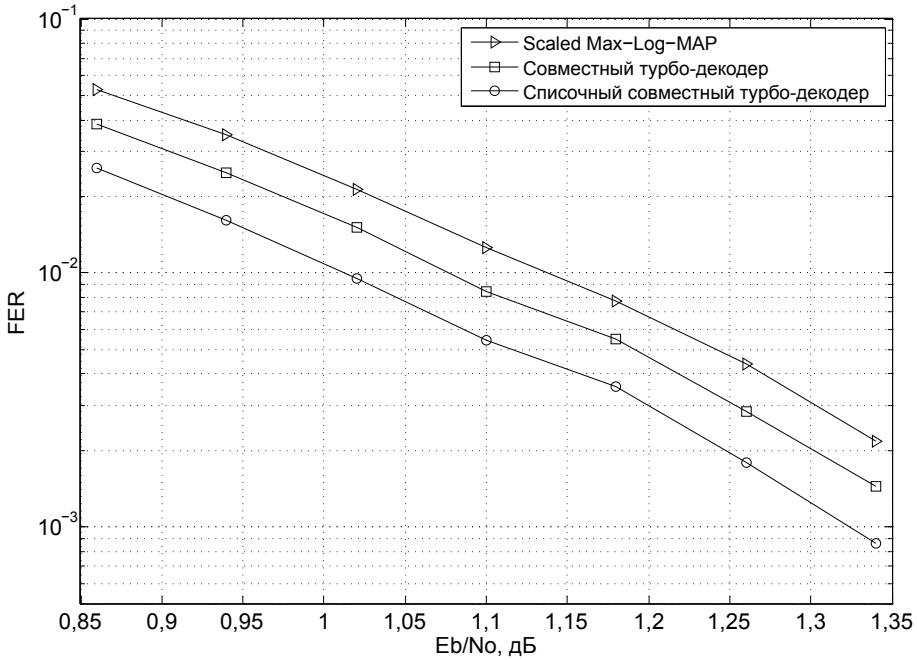

Рисунок 3.13 – Результаты моделирования совместного декодера для вокодера AMR-WB с учетом проверки CRC (FER)

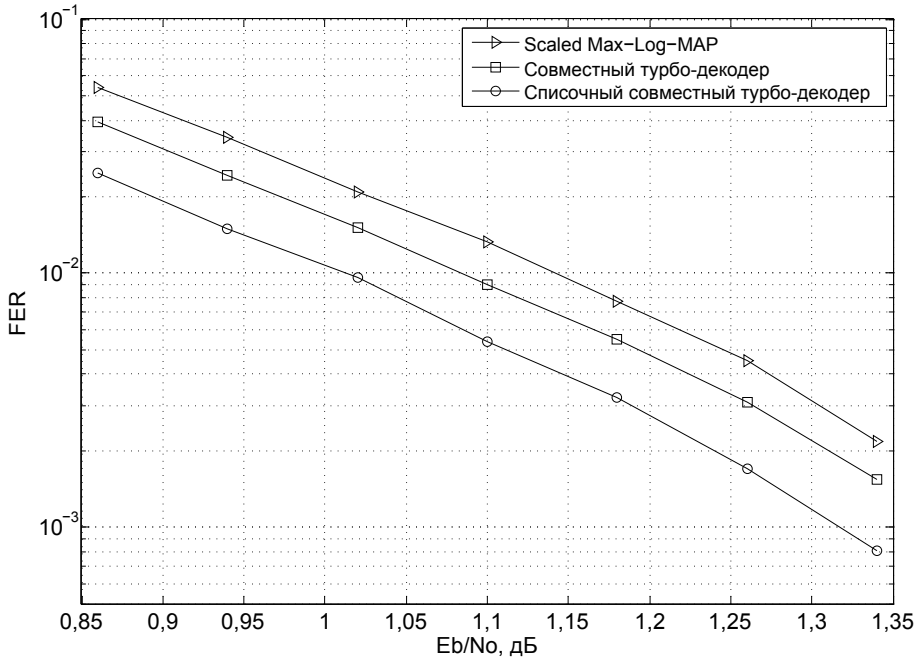

Рисунок 3.14 – Результаты моделирования совместного декодера для вокодера AMR-WB без учета проверки CRC (FER)

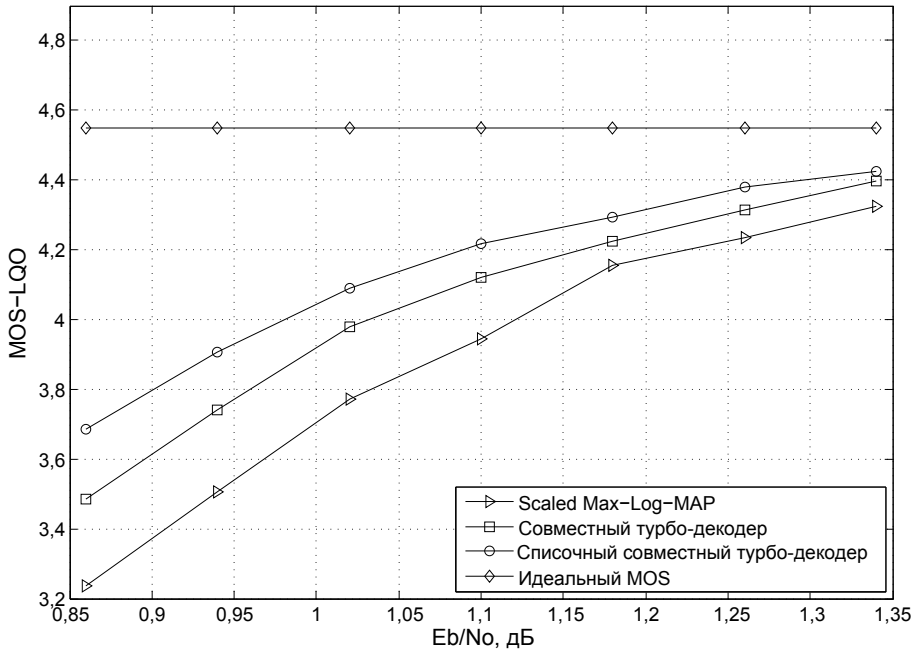

Рисунок 3.15 – Результаты моделирования совместного декодера для вокодера AMR-NB с учетом проверки CRC (MOS)

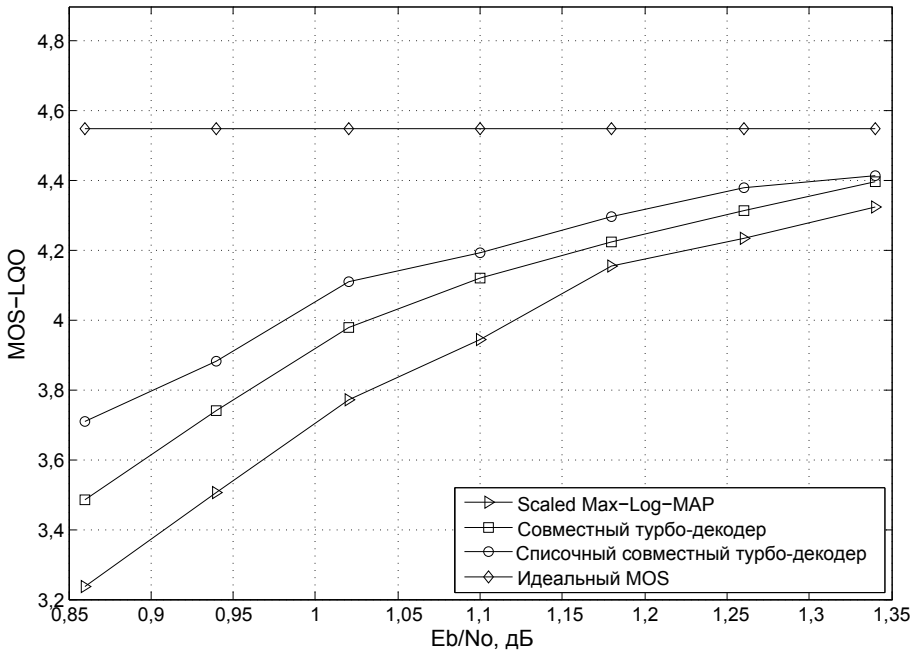

Рисунок 3.16 – Результаты моделирования совместного декодера для вокодера AMR-NB без учета проверки CRC (MOS)

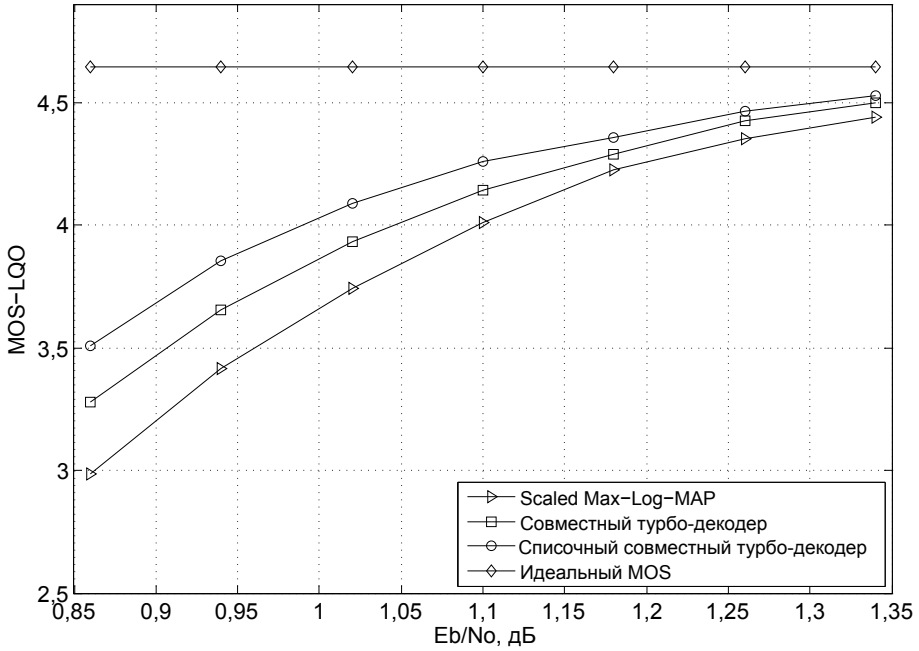

Рисунок 3.17 – Результаты моделирования совместного декодера для вокодера AMR-WB с учетом проверки CRC (MOS)

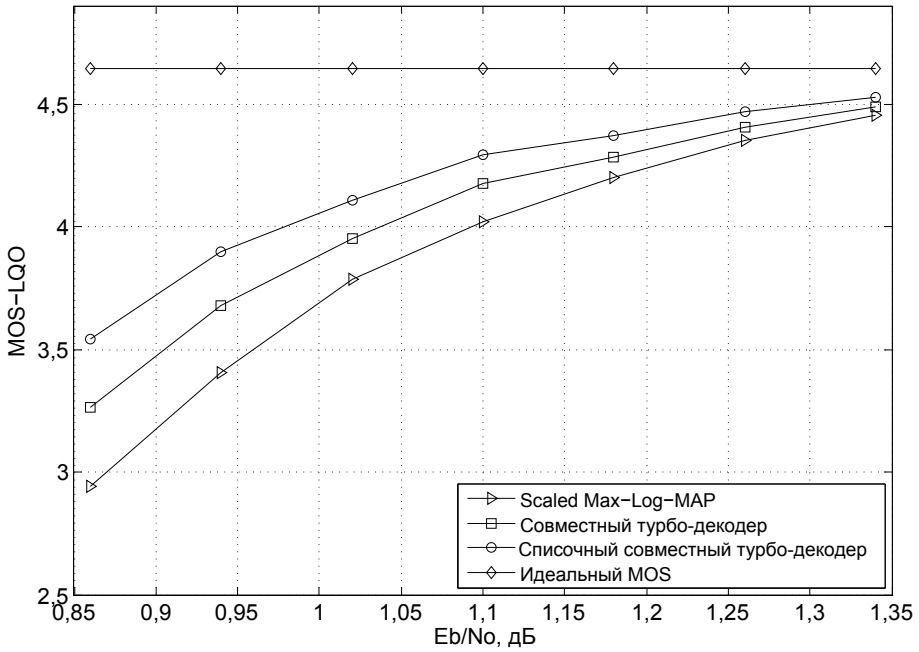

Рисунок 3.18 – Результаты моделирования совместного декодера для вокодера AMR-WB без учета проверки CRC (MOS)

4 Списочное декодирование турбокода для сети стандарта 3GPP LTE

4.1 Вводные замечания

Как было показано в разделе 3, использование совместного декодирования позволяет улучшить качество передачи речи в системе LTE за счет известной избыточности на выходе вокодера. Однако для различных режимов кодирования речи этой избыточности может быть не достаточно для получения интересных результатов. Использование универсального списочного декодера турбокода, который показывает стабильный прирост производительности на той или иной длине и не зависит от избыточности источника, является хорошим способом обеспечить требуемое качество передачи.

В данном разделе диссертационной работы произведен анализ возможного выигрыша, который может дать использование списочного декодера турбокода при передаче речевых данных. Анализ выполнен для нескольких режимов сжатия вокодеров AMR-NB и AMR-WB, в том числе и для основных, которые обсуждались в предыдущем разделе.

В разделе 4.2 дано описание системы LTE и передачи речи по протоколу IP в ней (VoLTE, Voice over LTE). В разделе 4.3 представлены результаты моделирования параллельного списочного декодера для системы VoLTE.

4.2 Передача голосовых данных в системе 3GPP LTE

Система 3GPP LTE на данный момент является одной из наиболее прогрессивных. Условно сеть LTE состоит из двух частей: E-UTRAN (Evolved Universal Terrestrial Radio Access Network) и EPC (Evolved Packet Core) (рисунок 4.1).

E-UTRAN является сетью радиодоступа и отвечает за обеспечение пере-

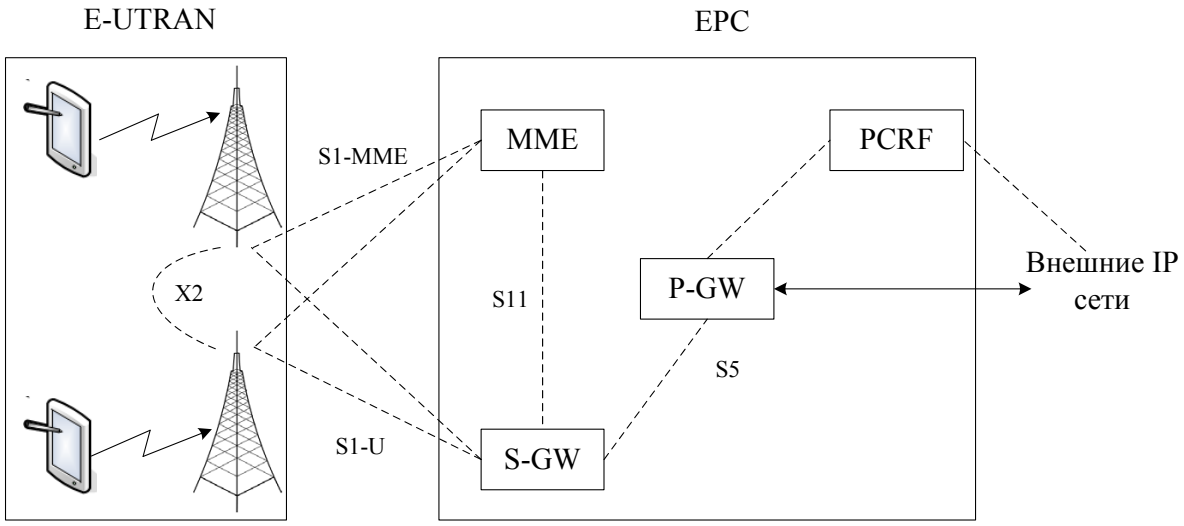

Рисунок 4.1 – Структура сети 3GPP LTE

дачи данных от пользователя и к нему, и состоит из базовой станции и мобильного устройства. EPC является опорной сетью оператора и включает в себя блок управления мобильностью (MME - Mobile Management Entity), шлюз передачи пакетов (P-GW - Packet Data Network Gateway), обслуживающий шлюз (S-GW - Serving Gateway) и узел установления счетов абонентам (PCRF - Policy and Charging Rules Function). Блок управления мобильностью обрабатывает все служебные сообщения (сетевую сигнализацию) между мобильным устройством и опорной сетью. Основными функциями MME является создание, поддержание и удаление потоков данных, контроль мобильности, управление безопасностью, авторизации и аутентификации мобильных устройств и другие служебные функции. Ключи шифрования, которые используются на PDCP уровне, также устанавливаются шлюзом MME. S-GW отвечает за маршрутизацию пакетов и переправляет пользовательские данные от подсистемы базовых станций, обслуживающих абонентов. Также шлюз S-GW выполняет функцию управления мобильностью для пользовательских данных при переходе абонентского устройства от одной базовой станции к другой. При передаче данных к пользователю S-GW производит пейджинг устройства, после чего следует сам процесс передачи. P-GW обеспечивает сообщение с внешними сетями, такими как Internet и др., при этом он также сообщается с узлом PCRF, для биллинга. Пользователь

может иметь несколько соединений с разными P-GW узлами, если требуется подключение к нескольким внешним сетям. Оба шлюза S-GW и P-GW поддерживают функцию узаконенного перехвата. PCRF отвечает за выставление счетов за предоставляемые услуги.

Передача голоса в стандарте 3GPP LTE может осуществляться двумя способами: IP телефония и перевод устройства в режим связи с коммутацией пакетов (GSM или UMTS). Последний вариант предусмотрен для более быстрого развертывания LTE сетей, т.е. в итоговом результате передача будет осуществляться по протоколу IP.

IP телефония в LTE происходить через инфраструктуру IMS, которая отвечает за предоставление основанных на протоколе IP мультимедиа услуг. Логически IMS расположена между транспортным уровнем и уровнем приложений, образуя прослойку между протоколами и приложениями, примерами которых являются IP/TCP/UDP и вокодер AMR, соответственно. IMS является внешней IP сетью для сети LTE, за которую отвечает отдельный P-GW. IMS использует интернет протоколы, стек которых показан на рисунке 4.2. Основными являются следующие протоколы:

1. SIP (Session Initiation Protocol) [100] - протокол для установления ,окончания и изменения параметров сеанса связи. Необходимо отметить, что SIP не отвечает за передачу мультимедиа данных.
2. SDP (Session Description Protocol) [101] - предназначен для описания сеанса IP телефонии, или другой сессии передачи мультимедиа.
3. RTP (Real-time Transport Ptotocol) [91] - протокол непосредственно для передачи голосовых или видео данных.

После того, как пользователь присоединился к сети LTE, получил IP адрес и установил соединение IMS, может начаться сеанс связи. При этом для защиты пакетов при передаче по беспроводному каналу используется шифрование данных на PDCP уровне, дополнительно шифрование SIP пакетов осуществляется по протоколу IPSec, для конфиденциальности голосовых данных стандарт

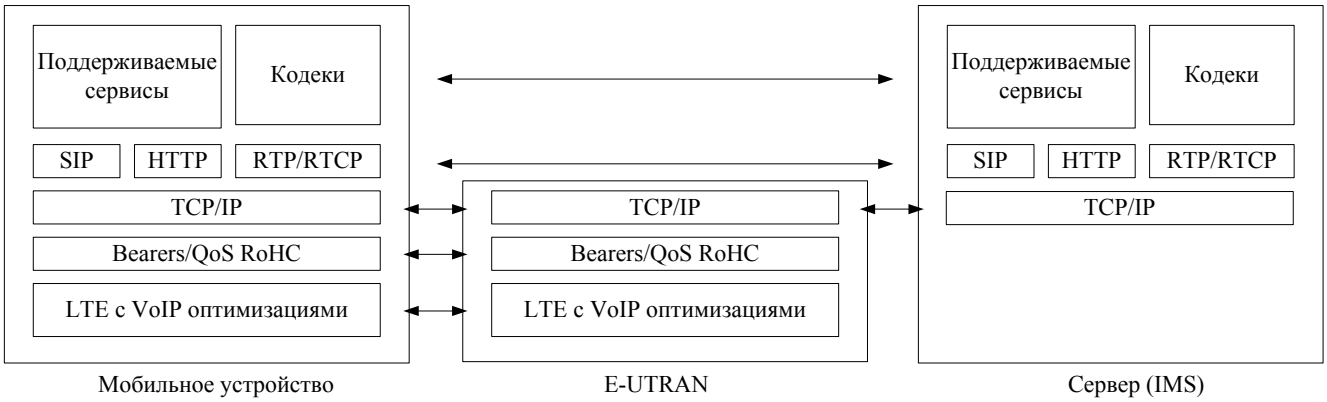

Рисунок 4.2 – Стек протоколов IMS

предусматривает передачу с помощью безопасного RTP протокола SRTP (Secure RTP), который шифрует непосредственно содержимое RTP пакета [102]. Принимая данные, базовая станция снимает шифр наложенный на PDCP уровне, после чего пакет передается к P-GW, который отвечает за соединение с сетью IMS.

На качество переданного голоса влияет как успешность приема на базовых станциях системы, так и на мобильных устройствах. В прошлом разделе был описан алгоритм совместного декодирования турбокода и источника, но как было сказано для внедрения такого алгоритма необходимо проводить глубокую межуровневую оптимизацию. Т.е. все уровни обработки данных должны быть прозрачными и доступными. Часто такой подход является не желательным и увеличивает задержку при обработке данных. В случае шифрования на PDCP уровне совместное декодирование может быть использовано, т.к. шифр работает в режиме гаммирования, однако это требует дополнительного усложнения приемника. Если передача данных происходит по протоколу SRTP, то использование совместного декодера просто не возможно. Следовательно, в некоторых случаях возможно лишь независимое улучшение на физическом уровне и использование параллельного списочного декодера выглядят более привлекательными.

4.3 Использование списочного декодера в VoLTE

Практика показывает, что в среднем участники телефонного разговора говорят лишь 50% времени. Выключая передачу в такие моменты, можно сохранить заряд батареи и обеспечить большую жизнь мобильного устройства. Пример такого разговора показан на рисунке 4.3 и, как видно, пока один абонент говорит, второй слушает. В 3GPP LTE это предусмотрено и используется алгоритм прерывистой передачи DTX, который во время пауз в разговоре отключает передатчик. Поэтому выходом алгоритма AMR являются пакеты трех типов: голосовые, с описанием комфортного шума и не содержащие данных. Последние говорят о том, что для синтеза речи должен быть использован последний успешно принятый пакет с данными. Такие пакеты не посылаются в канал, за счет чего происходит экономия заряда батареи. Анализ потока AMR показал, что при использовании DTX передача происходит всплесками: передается череда голосовых пакетов, после чего следует пакет с описанием комфортного шума и передача прекращается на какое-то время.

Для декодирования турбокода был использован параллельный списочный декодер турбокода со списком $L = 16$. Для каждого кодека были рассмотрены два основных режима работы: для AMR-NB 10,2 Кбит/с и 12,2 Кбит/с, для AMR-WB 8,85 Кбит/с и 12,65 Кбит/с. Режимы вокодеров рассматривались отдельно друг от друга. Схема системы, для которой проводилось моделирование аналогична показанной на рисунке 3.10. Отличие проведенного моделирования заключалось в том, что пакеты с информацией о комфортном шуме также передавались по каналу, а не считались принятыми успешно. Для оценки улучшения качества принимаемой речи была использована метрика MOS-LQO, которая измерялась с помощью алгоритма PESQ. Так как в качестве канала использовался канал с АБГШ, то графики MOS построены в зависимости от E_b/N_0.

Как видно на рисунках 4.4 и 4.5, параллельный списочный декодер обеспечивает улучшение качество принимаемой речи на 0,18-0,5 MOS для AMR-NB

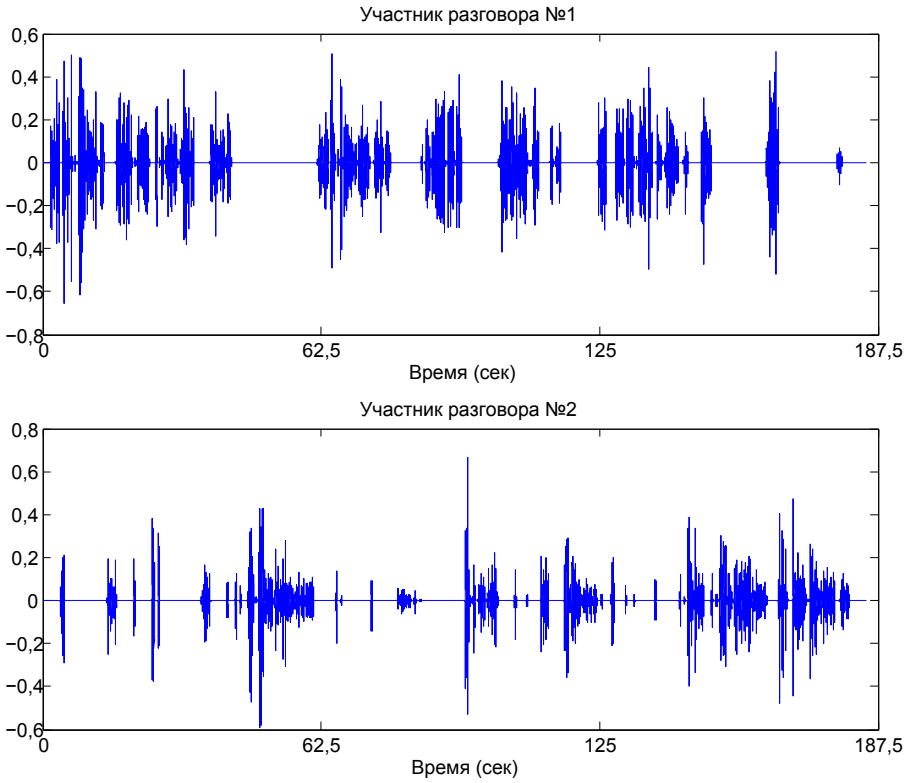

Рисунок 4.3 – Пример телефонного разговора. Сигнал взят из [103]

в режиме 12,2 Кбит/с и на 0,18-0,45 MOS для режима 10,2 Кбит/с в зависимости от значений E_b/N_0. Улучшение для AMR-WB составляет 0,15-0,5 MOS и 0,2-0,6 MOS для режимов 12,65 Кбит/с и 8,85 Кбит/с, соответственно.

Стоит отметить, что возможно регулировать сложность декодера за счет оценки качества голоса на приемной стороне. Для данной цели используют пассивные алгоритмы, для которых не требуется знание переданного потока [4, 8]. В этом случае в зависимости от полученной оценки можно варьировать размерность списка, что приведет к уменьшению энергозатрат на декодирование. В [4,8] предложен эффективный алгоритм, с помощью которого на базовой станции можно производить такие измерения. Изменение параметров декодирования необходимо рассматривать в зависимости от требования к качеству обслуживания пользователей в конкретной системе.

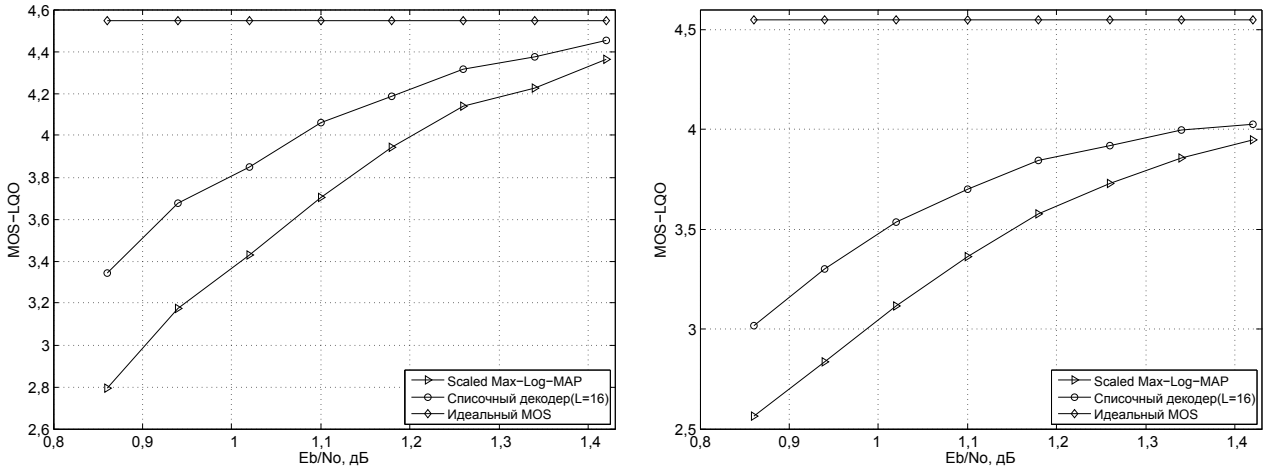

Рисунок 4.4 – Результаты моделирования списочного турбо-декодера для системы VoLTE. Режимы вокодера AMR-NB 12,2 кбит/с (слева), 10,2 кбит/с (справа)

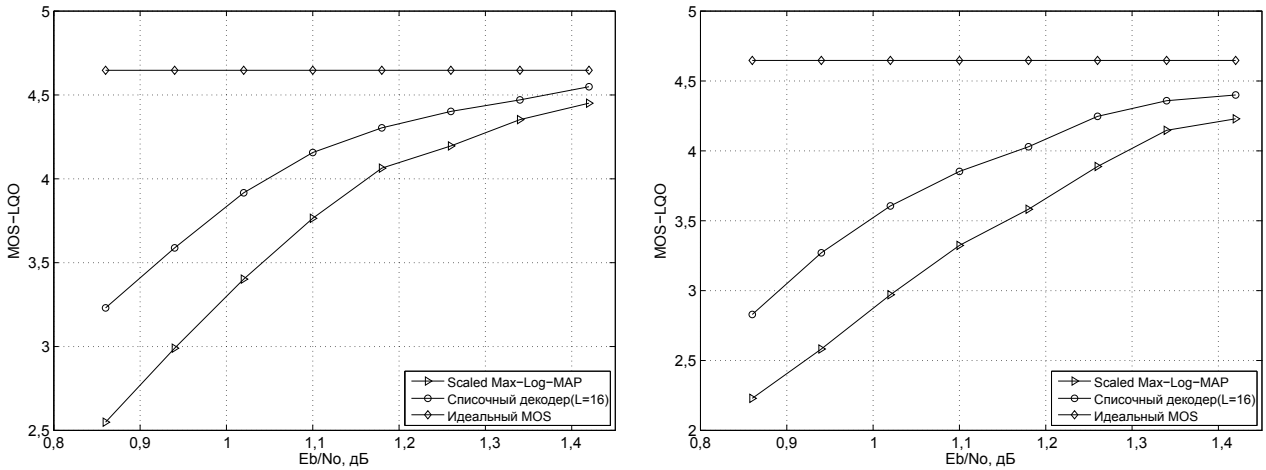

Рисунок 4.5 – Результаты моделирования списочного турбо-декодера для системы VoLTE. Режимы вокодера AMR-WB 12,65 кбит/с (слева), 8,85 кбит/с (справа)

4.4 Заключение и выводы по разделу

В данном разделе показано улучшение качества принимаемой речи, достигаемое при использовании предложенного во втором разделе списочного декодера турбокода в стандарте 3GPP LTE. Рассмотрены основные режимы работы вокодеров AMR-NB и AMR-WB, которые используются для передачи голоса в сети. Проанализирована возможность использования совместного декодирования в тех или иных ситуациях в сети LTE. Сделан вывод, что при некоторых режимах передачи голосовых пакетов и невозможности проведения глубокой

межуровневой оптимизации при проектировании приемника совместное декодирование кодов канала и источника не возможно. В этом случае целесообразным является использование параллельного списочного декодера. Показано, что списочное декодирование в среднем позволяет улучшить качество принимаемой речи минимум на 0,15-0,5 MOS в зависимости от значения E_b/N_0 и перевести качество голоса из класса «удовлетворительно» в класс «хорошо»

Заключение

В диссертационной работе рассмотрена обработка информации с помощью турбокодов. Основное внимание уделено улучшению параметров декодирования турбокодов, как единственному способу улучшения параметров помехоустойчивого кодирования в существующих стандартах связи. Рассмотрены два подхода к улучшению декодера: списочное декодирование и совместное декодирование кодов канала и источника. Для обоих подходов предложены алгоритмы, которые позволяют улучшить качество приема информации и учитывают специфику передачи в современных мобильных сетях, таких как 3GPP LTE. Для анализа разработанных алгоритмов проведено моделирование в канале с АБГШ. Так как голосовые данные являются одним из основных типов данных в сотовых сетях, то для иллюстрации эффективности предложенных алгоритмов приведены оценки улучшения качества принимаемой речи для вокодеров AMR-NB и AMR-WB в системе 3GPP LTE.

Основные результаты диссертационной работы могут быть сформулированы следующим образом:

1. Проведено исследование списочных декодеров турбокодов. Показано, что существующие списочные алгоритмы дают уменьшение вероятности ошибки на пакет по сравнению с классическим турбо-декодером лишь на малых длинах информационных слов. При увеличении длины слова эффективность рассмотренных алгоритмов резко снижается, и выигрыш становится незначительным, либо рост сложности алгоритма приводит к невозможности использования декодера.
2. Предложен алгоритм списочный декодирования турбокода, который позволяет уменьшить вероятность ошибки на пакет по сравнению с классическим турбо-декодером на 0,15—0,4 дБ в канале с АБГШ. При этом задержка декодирования алгоритма приемлема для его практического использования в сетях мобильной связи и близка к задержке классического

турбо-декодера.

3. Предложена оконная модификация списочного декодера турбокода, которая позволяет ускорить процесс генерации списка и уменьшить задержку декодирования.
4. Разработан совместный декодер турбокода и кода источника, учитывающий наличие значительного числа случайных бит в информационном слове, которые не могут быть использованы для совместного декодирования и которые значительно снижают эффективность совместного декодирования в существующих подходах.
5. Разработан совместный декодер турбокода стандарта 3GPP LTE и вокодеров AMR-NB и AMR-WB. Проведен анализ избыточности данных на выходе вокодеров AMR-NB и AMR-WB для русской речи. Показано, что предложенный декодер уменьшает вероятность ошибки на пакет по сравнению с классическим турбо-декодером на 0,15 дБ. Улучшение качества принимаемой речи при этом варьируется от 0,15 до 0,4 пунктов по шкале MOS.
6. Проанализированы сценарии, при которых целесообразно использовать списочное и совместное декодирование. Рассмотрено применение предложенного списочного декодера турбокода для улучшения качества принимаемой речи в сетях связи стандарта 3GPP LTE. Показано, что списочный декодер позволяет улучшить качество речи на приемнике на 0,15—-0,5 пунктов по шкале MOS по сравнению с классическим турбо-декодером.

Список литературы

1. *Акмалходжаев, А. И.* Совместный списочный декодер турбокода и вокодера AMR-NB для сетей четвертого поколения / А. И. Акмалходжаев // *Информационно-управляющие системы*. — 2014. — Т. 69, № 2. — С. 63–70.
2. *Акмалходжаев, А. И.* Новый алгоритм списочного декодирования турбокодов / А. И. Акмалходжаев, А. В. Козлов // *Известия высших учебных заведений. Приборостроение*. — 2013. — Август. — Т. 56, № 8. — С. 20–23.
3. *Акмалходжаев, А. И.* Списочное декодирование турбокодов / А. И. Акмалходжаев, А. В. Козлов // *СПИСОК-2012. Материалы всероссийской научной конференции по проблемам информатики*. — 2012. — С. 194–199.
4. *Акмалходжаев, А. И.* Новый пассивный метод оценки качества голоса в сети 3GPP LTE / А. И. Акмалходжаев // *СПИСОК-2014. Материалы всероссийской научной конференции по проблемам информатики*. — 2014. — С. 215–223.
5. *Акмалходжаев, А. И.* Совместное декодирование канала и источника / А. И. Акмалходжаев // *Научная сессия ГУАП: сборник докладов*. — 2013. — Т. 1. — С. 71–75.
6. *Akmalkhodzhaev, A. I.* New Iterative Turbo Code List Decoder / A. I. Akmalkhodzhaev, A. V. Kozlov // *XIV International Symposium "Problems of Redundancy in Information and Control Systems"*. — 2014. — P. 15–18.
7. *Akmalkhodzhaev, A. I.* Joint decoding of turbo code and AMR-WB vocoder in 3GPP LTE system / A. I. Akmalkhodzhaev // *6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. — 2014. — P. 402–406.
8. *Akmalkhodzhaev, A. I.* New Non-intrusive Speech Quality Assessment Algorithm for Wireless Networks / A. I. Akmalkhodzhaev, A. V. Kozlov // *Intelligent Interactive Multimedia Systems and Services*. — 2015. — P. 215–225.

9. *Акмалходжаев, А. И.* Сравнительны анализ алгоритмов декодирования турбокодов на примере турбокода стандарта 3GPP LTE / А. И. Акмалходжаев // *Вестник Таджикского технического университета*. — 2014. — Т. 28, № 4. — С. 44–47.
10. *Berrou, C.* Near Shannon limit error-correcting coding: turbo codes. / C. Berrou, A. Glavieux, P. Thitimajshima // *Proceedings of the IEEE International Conference on Communications (ICC 93)*. — 1993. — May. — Vol. 2. — P. 1064–1070.
11. Optimal decoding of linear codes for minimizing symbol error rate / L. Bahl, J. Cocke, F. Jelinek, J. Raviv // *IEEE Transactions on Information Theory*. — 1974. — March. — Vol. 20, no. 2. — P. 284–287.
12. *Hagenauer, J.* A viterbi algorithm with soft-decision outputs and its applications / J. Hagenauer, P. Hoeher // *Proc. IEEE GLOBECOM '89*. — 1989. — November. — Vol. 3. — P. 1680–1686.
13. *Robertson, P.* A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain / P. Robertson, E. Villebrun, P. Hoeher // *Proceedings of the IEEE International Conference on Communications (ICC 95)*. — 1995. — June. — Vol. 2. — P. 1009–1013.
14. *Divsalar, D.* On the design of turbo codes / D. Divsalar, F. Pollara // *The Telecommunications and Data Acquisition Progress Report, TDA PR 42-123*. — 1995. — November. — P. 99–121.
15. *Divsalar, D.* Weight distributions for turbo codes using random and nonrandom permutations / D. Divsalar, S. Dolinar // *The Telecommunications and Data Acquisition Progress Report, TDA PR 42-122*. — 1995. — August. — P. 56–65.
16. *Зяблов, В.В.* Дистанционные свойства турбо кодов с различными перемежителями / В.В. Зяблов, М.А. Цветков // *Информацинные процессы*. — 2003. — Т. 3, № 2. — С. 83–96.
17. *Chatzigeorgiou, I. A.* Performance Analysis and Design of Punctured Turbo Codes: Ph.D. thesis / University of Cambridge. Department of Engineering. — 2006. — P. 150.

18. *Divsalar, D.* Multiple turbo codes / D. Divsalar, F. Pollara // *IEEE Military Communications Conference (MILCOM '95)*. — 1995. — November. — Vol. 1. — P. 279–285.

19. *Divsalar, D.* Multiple turbo codes for deep-space communications / D. Divsalar, F. Pollara // *PL TDA Progress Report 42-121*. — 1995. — May. — P. 66–77.

20. *Heim, A.* Turbo decoding: Why stopping-criteria do work / A. Heim, U. Sorger // *5th International Symposium on Turbo Codes and Related Topics*. — 2008. — September. — P. 255–259.

21. *Moon, T. K.* Error Correction Coding. Mathematical Methods and Algorithms / T. K. Moon. — A John Wiley and Sons, Inc., 2005. — P. 800.

22. *Brink, S.* Convergence behavior of iteratively decoded parallel concatenated codes / S. Brink // *IEEE Transactions on Communications*. — 2001. — October. — Vol. 49, no. 10. — P. 1727–1737.

23. *Matache, A.* Stopping rules for turbo decoders / A. Matache, S. Dolinar, F. Pollara // *TMO Progress Report 42-142*. — 2000. — August. — P. 1–22.

24. *Gilbert, F.* Low complexity stopping criteria for UMTS turbo decoders / F. Gilbert, F. Kienle, N. Wehn // *The 57'th IEEE Semiannual Vehicular Technology Conference*. — 2003. — April. — Vol. 4. — P. 2376–2380.

25. *Shao, R. Y.* Two simple stopping criteria for turbo decoding / R. Y. Shao, S. Lin, M. P. C. Fossorier // *IEEE Transactions on Communications*. — 1999. — Vol. 47, no. 8. — P. 1117–1120.

26. *Zhang, J.* Shuffled iterative decoding / J. Zhang, M. P. C. Fossorier // *IEEE Transactions on Communications*. — 2005. — February. — Vol. 53, no. 2. — P. 209–213.

27. Replica shuffled iterative decoding / J. Zhang, Y. Wang, M. P. C. Fossorier, J.S. Yedidia // *Proceedings of the 2005 International Symposium on Information Theory*. — 2005. — September. — P. 454–458.

28. *Zhang, J.* Iterative decoding with replicas / J. Zhang, M. P. C. Fossorier, J.S. Yedidia // *IEEE Transactions on Information Theory*. — 2007. — May. — Vol. 53, no. 5. — P. 1644–1663.

29. *Johannesson, R.* Fundamentals of Convolutional Coding / R. Johannesson, K. Zigangirov. — Wiley-IEEE Press, 1999. — P. 442.

30. *Krouk, E.* Modulation and Coding Techniques in Wireless Communications / E. Krouk, A. Ovchinnikov, J. Poikonen. — John Wiley and Sons, 2011. — P. 680.

31. *Viterbi, A. J.* Error bounds for convolutional codes and an asymptotically optimum decoding algorithm / A. J. Viterbi // *IEEE Transactions on Information Theory*. — 1967. — April. — Vol. 13, no. 2. — P. 260–269.

32. *G. D. Forney Jr.* The Viterbi algorithm / G. D. Forney Jr. // *Proceedings of the IEEE*. — 1973. — March. — Vol. 61. — P. 268–278.

33. *Kamuf, M.* Trellis Decoding. From Algorithm to Flexible Architectures / M. Kamuf. — Department of Electroscience: Sweden, Lund, Tryckeriet i E-huset, 2007. — P. 128.

34. *Fossorier, M. P. C.* Bi-directional SOVA decoding for turbo-codes / M. P. C. Fossorier, S. Lin, C. Xu // *IEEE Communications Letters*. — 2000. — December. — Vol. 4, no. 12. — P. 405–407.

35. *Hanzo, L.* Turbo Coding, Turbo Equalisation and Space-Time Coding for Transmission over Fading Channels / L. Hanzo, T.H. Liew, B.L. Yeap. — Wiley-IEEE Press, 2002. — P. 766.

36. *Glavieux, A.* Channel Coding in Communication Networks: From Theory to Turbocodes / A. Glavieux. — Wiley-ISTE, 2007. — P. 418.

37. *Johnson, S. J.* Iterative Error Correction. Turbo, Low-Density Parity-Check and Repeat–Accumulate Codes / S. J. Johnson. — Cambridge University Press, 2009. — December. — P. 356.

38. *Schlegel, C. B.* Trellis and Turbo Coding / C. B. Schlegel, L. C. Perez. — Wiley-IEEE Press, 2003. — P. 300.

39. On error floor and free distance of turbo codes / R. Garello, F. Chiaraluce, P. Pierleoni et al. // *IEEE International Conference on Communications (ICC 2001)*. — 2001. — June. — Vol. 1. — P. 45–49.

40. *Richardson, T.* The geometry of turbo-decoding dynamics / T. Richardson // *IEEE Transactions on Information Theory*. — 2000. — January. — Vol. 46. — P. 9–23.

41. *Hagenauer, J.* Iterative decoding of binary block and convolutional codes / J. Hagenauer, E. Offer, L. Papke // *IEEE Transactions on Information Theory*. — 1996. — Vol. 42, no. 2. — P. 429–445.

42. Reduced latency turbo decoding / J. Zhang, Y. Wang, M. P. C. Fossorier, J.S. Yedidia // *IEEE 6th Workshop on Signal Processing Advances in Wireless Communications*. — 2005. — June. — P. 930–934.

43. *Claussen, H.* Improved max-log-map turbo decoding by maximization of mutual information transfer / H. Claussen, H. R. Karimi, B. Mulgrew // *EURASIP Journal on Applied Signal Processing*. — 2005. — January. — P. 820–827.

44. *Taskaldiran, M.* A comparative study on the modified max-log-map turbo decoding by extrinsic information scaling / M. Taskaldiran, R. C. S. Morling, I. Kale // *Wireless Telecommunications Symposium*. — 2007. — April. — P. 1–5.

45. 3GPP TS 36.212. 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (Release 9) / 3GPP. — Valbonne, France, 2011-09-27. — P. 60.

46. *Narayanan, K. R.* List decoding of turbo codes / K. R. Narayanan, G. L. Stueber // *IEEE Transactions on Communications*. — 1998. — June. — Vol. 46, no. 6. — P. 754–762.

47. *Sadowsky, J. S.* A maximum likelihood decoding algorithm for turbo codes / J. S. Sadowsky // *IEEE Global Telecommunications Conference, GLOBTELECOM '97*. — 1997. — November. — Vol. 2. — P. 929–933.

48. *Gallager, R. G.* Low-Density Parity-Check codes / R. G. Gallager. — Massachusetts Institute of Technology, 1963. — P. 90.

49. *Elias, P.* List decoding for noisy channels / P. Elias // *1957-IRE WESCON Convention Record*. — 1957. — September. — Vol. 2. — P. 94–104.

50. *Guruswami, V.* List Decoding of Error-Correcting Codes: Ph.D. thesis / Massachusetts Institute of Technology. — 2001. — September. — P. 315.

51. *Sudan, M.* List decoding: algorithms and applications / M. Sudan // *SIGACT NEWS*. — 2000. — March. — Vol. 31, no. 1. — P. 16–27.

52. *Leanderson, C. F.* On list sequence turbo decoding / C. F. Leanderson, C.-E. W. Sundberg // *IEEE Transactions on Communications*. — 2005. — May. — Vol. 53, no. 5. — P. 760–763.

53. *Seshadri, N.* List Viterbi decoding algorithms with applications / N. Seshadri, C.-E. W. Sundberg // *IEEE Transactions on Communications*. — 1994. — February/March/April. — Vol. 42, no. 2/3/4. — P. 313–323.

54. *Poulliat, C.* Efficient decoding of turbo codes with nonbinary belief propagation / C. Poulliat, D. Declercq, T. Lestable // *EURASIP Journal onWireless Communications and Networking*. — 2008. — April. — P. 1–10.

55. *Овчинников, А.А.* Обработка информации при передаче LPDC-кодами по дискретным и полунепрерывним каналам : дис. ..... канд. тех. наук : 05.13.01 / А.А. Овчинников. — СПб.: ГУАП, 2004. — С. 141.

56. A CRC-aided hybrid decoding algorithm for turbo codes / W. Yuejun, M. Jiang, B. Xia et al. // *IEEE wireless communications letters*. — 2013. — October. — Vol. 2, no. 5. — P. 471–474.

57. *Nill, C.* List and soft symbol output viterbi algorithms: Extensions and comparisons / C. Nill, C.-E. W. Sundberg // *IEEE Transactions on Communications*. — 1995. — Vol. 43, no. 2/3/4. — P. 277–287.

58. *Refaey, A.* A new approach for FEC decoding based on the BP algorithm in LTE and WiMAX systems / A. Refaey, S. Roy, P. Fortier // *12th Canadian Workshop on Information Theory (CWIT)*. — 2011. — May. — P. 9–14.

59. *Sujatha, K.* A novel approach for FEC decoding based on the BP algorithm in LTE and WiMAX systems / K. Sujatha, S. Katteda, S. Babu // *International Journal of Engineering Research and Development*. — 2012. — december. — Vol. 5. — P. 6–13.

60. *Poulliat, C.* Preprocessing for an efficient decoding of turbo-codes with non-binary belief propagation / C. Poulliat, D. Declercq, T. Lestable // *5th International Symposium onTurbo Codes and Related Topics*. — 2008. — September. — P. 362–367.

61. *Скляр, Б.* Цифровая связь. Теоретические основы и практическое применение / Б. Скляр. — Вильямс, 2007. — С. 1104.

62. *Benedetto, S.* Algorithm for continuous decoding of turbo codes / S. Benedetto, D. Divsalar G. Montorsi, F. Pollara // *IEEE Electronics Letters*. — 1996. — February. — P. 314–315.

63. *Sun, Y.* Efficient hardware implementation of a highly-parallel 3GPP LTE/LTE-advance turbo decode / Y. Sun, J. R. Cavallaro // *INTEGRATION, the VLSI journal 44*. — 2011. — P. 305–315.

64. *Robertson, P.* A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain / P. Robertson, E. Villebrun, P. Hoeher // *IEEE International Conference on Communications (ICC '95)*. — 1995. — June. — Vol. 2. — P. 1009–1013.

65. *Shannon, C. E.* A mathematical theory of communication / C. E. Shannon // *The Bell System Technical Journal*. — 1948. — Vol. 27. — P. 379–423.

66. *Duhamel, P.* Joint Source-Channel decoding. A Cross Layer Perspective with Applications in Video Broadcasting over Mobile and Wireless Networks / P. Duhamel, M. Kieffer. — A John Wiley and Sons, Inc., 2010. — January. — P. 334.

67. *Phambo, N.* Optimal detection of discrete markov sources over discrete memoryless channels-applications to combined source-channel coding / N. Phambo,

N. Farvardin // *IEEE Transactions on Information Theory*. — 1994. — Jan. — Vol. 40. — P. 186–193.

68. *Cover, T. M.* Elements of Information Theory / T. M. Cover, J. A. Thomas. — A John Wiley and Sons, Inc., 2006. — July. — P. 776.
69. *Perkert, R.* Iterative source and channel decoding for GSM / R. Perkert, M. Kaindl, T. Hindelang // *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01)*. — 2001. — Vol. 4. — P. 2649–2652.
70. Application of a joint source-channel decoding technique to UMTS channel codes and OFDM modulation / M. Jeanne, I. Siaud, O. Seller, P. Siohan // *Proceedings of 11'th International Conference on Telecommunications*. — 2004. — August. — P. 914–923.
71. *Hindelang, T.* Combined source/channel (de-)coding: Can a priori information be used twice? / T. Hindelang, T. Fingscheidt, N. Seshadri and R. V. Cox // *IEEE International Conference on Communications, 2000. ICC 2000*. — 2000. — Vol. 3. — P. 1208–1212.
72. *Adrat, M.* Iterative source-channel decoder using extrinsic information from softbit-source decoding / M. Adrat, P. Vary, J. Spittka // *IEEE International Conference on Acoustics, Speech, and Signal Processing*. — 2001. — Vol. 4. — P. 2653–2656.
73. 3GPP TS 26.071. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Mandatory speech codec speech processing functions; AMR speech codec; General description (Release 9). — Valbonne, France: 3GPP, 2009-12-18. — P. 12.
74. 3GPP TS 26.171. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Speech codec speech processing functions; Adaptive Multi-Rate - Wideband (AMR-WB) speech codec; General description (Release 9). — Valbonne, France: 3GPP, 2009-12-18. — P. 12.

75. 3GPP TS 26.094. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Mandatory speech codec speech processing functions; Adaptive Multi-Rate (AMR) speech codec; Voice Activity Detector (VAD) (Release 9). — Valbonne, France: 3GPP, 2009-12-18. — P. 25.
76. 3GPP TS 26.194 . 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Speech codec speech processing functions; Adaptive Multi-Rate - Wideband (AMR-WB) speech codec; Voice Activity Detector (VAD) (Release 9). — Valbonne, France: 3GPP, 2009-12-18. — P. 16.
77. 3GPP TS 26.092. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Mandatory speech codec speech processing functions; Adaptive Multi-Rate (AMR) speech codec; Comfort noise aspects (Release 9). — Valbonne, France: 3GPP, 2009-12-18. — P. 12.
78. 3GPP TS 26.192. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Speech codec speech processing functions; Adaptive Multi-Rate - Wideband (AMR-WB) speech codec; Comfort noise aspects (Release 9). — Valbonne, France: 3GPP, 2009.
79. 3GPP TS 26.091. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Mandatory Speech Codec speech processing functions; Adaptive Multi-Rate (AMR) speech codec; Error concealment of lost frames (Release 9). — Valbonne, France: 3GPP, 2009-12-18. — P. 13.
80. 3GPP TS 26.191. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Speech codec speech processing functions; Adaptive Multi-Rate - Wideband (AMR-WB) speech codec; Error concealment of erroneous or lost frames (Release 9). — Valbonne, France: 3GPP, 2009-12-18. — P. 14.
81. *Рабинер, Л.* Цифровая обработка речевых сигналов / Л. Рабинер, Р. Шафер. — М.: Радио и Связь, 1981. — С. 496.

82. *Vary, P.* Digital Speech Transmission. Enhancement, Coding and Error Concealment / P. Vary, R. Martin. — John Wiley and Sons, Ltd, 2006. — March. — P. 644.
83. 3GPP TS 26.090. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Mandatory Speech Codec speech processing functions; Adaptive Multi-Rate (AMR) speech codec; Transcoding functions (Release 9). — Valbonne, France: 3GPP, 2011-09-30. — P. 55.
84. *Itakura, F.* Line spectrum representation of linear predictive coefficients of speech signals / F. Itakura // *J. Acoust. Soc. Am. 57, S35*. — 1975.
85. *Kondoz, A. M.* Digital Speech. Coding for Low Bit Rate Communication Systems. Second Edition / A. M. Kondoz. — John Wiley and Sons, 2004. — P. 460.
86. *Bistritz, Y.* Immittance spectral pairs (isp) for speech encoding / Y. Bistritz, S. Pellerm // *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. — 1993. — Vol. 2. — P. 9–12.
87. 3GPP TS 26.190. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Speech codec speech processing functions; Adaptive Multi-Rate - Wideband (AMR-WB) speech codec; Transcoding functions (Release 9). — Valbonne, France: 3GPP, 2009-12-18. — P. 51.
88. 3GPP TS 26.101. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Mandatory speech codec speech processing functions; Adaptive Multi-Rate (AMR) speech codec frame structure (Release 9). — Valbonne, France: 3GPP, 2009-12-18. — P. 20.
89. 3GPP TS 26.201. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Speech codec speech processing functions; Adaptive Multi-Rate - Wideband (AMR-WB) speech codec; Frame structure (Release 9). — Valbonne, France: 3GPP, 2009-12-18. — P. 23.
90. RTP payload format and file storage format for the adaptive multi-rate (AMR) and adaptive multi-rate wideband (AMR-WB) audio codecs / J. Sjoberg, M. Westerlund, A. Lakaniemi, Q. Xie // *RFC 4867*. — 2007. — April. — P. 59.

91. RTP: A Transport Protocol for Real-Time Applications / H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson // *RFC 3550*. — 2003. — P. 104.

92. *Postel, J.* User datagram protocol / J. Postel // *RFC 768*. — 1980. — August. — P. 3.

93. RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed / C. Bormann, C. Burmeister, M. Degermark et al. // *RFC 3095*. — 2001. — July. — P. 168.

94. *Fingscheidt, T.* Softbit speech decoding: A new approach to error concealment / T. Fingscheidt, P. Vary // *IEEE Transactions on Speech and Audio Processing*. — 2001. — March. — Vol. 9, no. 3. — P. 240–251.

95. 3GPP TS 36.213. 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures (Release 9). — Valbonne, France: 3GPP, 2010-10-03. — P. 80.

96. ITU-T Recommendation P.800. Series P: Methods for objective and subjective assessment of quality. Methods for objective and subjective assessment of quality. — ITU-T, 1996-08. — P. 37.

97. ITU-T Recommendation P.862. Series P: Methods for objective and subjective assessment of quality. Perceptual evaluation of speech quality(PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs / ITU-T. — ITU-T, 2001-02. — P. 30.

98. ITU-T Recommendation P.862.2. Series P: Methods for objective and subjective assessment of quality. Wideband extension to Recommendation P.862 for the assessment of wideband telephone networks and speech codecs / ITU-T. — ITU-T, 2007-11. — P. 12.

99. ITU-T Recommendation P.862.1. Series P: Methods for objective and subjective assessment of quality. Mapping function for transforming P.862 raw result scores to MOS-LQO / ITU-T. — ITU-T, 2003-11. — P. 12.

100. SIP: Session Initiation Protocol / J. Rosenberg, H. Schulzrinne, G. Camarillo et al. // *RFC 3261*. — 2002. — P. 269.

101. *Handley, M.* SDP: Session Description Protocol / M. Handley, V. Jacobson // *RFC 2327*. — 1998. — P. 42.

102. The Secure Real-time Transport Protocol (SRTP) / M. Baugher, D. McGrew, M. Naslund et al. // *RFC 3711*. — 2004. — P. 56.

103. ITU-T Recommendation P.561. Series P: Objective measuring apparatus. Appendix III: Digital speech recordings / ITU-T. — ITU-T, 1998-02. — P. 7.