УДК 621.396

Декодирование кодов LDPC с регулярной структурой проверочной матрицы способом Bit Flip

Копылов Д. А.[1], **Мендельсон М. А.**[1], **Егоров В. А.**[1], **Лютин В. И.**[2]

[1]Санкт-Петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича, Санкт-Петербург, 193232, Российская Федерация
[2]Военно-воздушная академия им. проф. Н. Е. Жуковского и Ю. А. Гагарина, Воронеж, 394064, Российская Федерация

Постановка задачи: *в современных системах связи большую популярность приобрели низкоплотностные коды LDPC, позволяющие реализовать на практике декодеры для них. В условиях жестких ограничений производительности оборудования актуальной является задача поиска и исследования характеристик алгоритмов декодирования этих кодов.* ***Целью работы*** *является нахождение упрощенного алгоритма декодирования кодов LDPC и исследование его помехоустойчивости.* ***Используемые методы:*** *для решения поставленной задачи используется имитационное моделирование на компьютере с целью сравнения помехоустойчивости известных алгоритмов декодирования и предложенного алгоритма.* ***Научная новизна:*** *исследованы характеристики помехоустойчивости представленного варианта алгоритма Bit Flip (с инверсией бит) для декодирования кода LDPC с регулярной структурой проверочной матрицы с применением градиентного спуска и процедурой выхода из локального минимума.* ***Результат:*** *найден конкретный вариант алгоритма Bit Flip для кода LDPC с регулярной структурой проверочной матрицы с применением градиентного спуска и процедурой выхода из локального минимума. Приведены результаты моделирования на компьютере.* ***Теоретическая значимость:*** *исследованы характеристики помехоустойчивости для предложенного варианта алгоритма декодирования кода LDPC с регулярной структурой проверочной матрицы.* ***Практическая значимость:*** *при применении в приемнике данного алгоритма декодирования кода LDPC появляется возможность по имеющимся зависимостям коэффициента ошибок от отношения сигнал/шум рассчитывать параметры системы связи.*

Ключевые слова: *помехоустойчивые коды, код LDPC, проверочная матрица, Bit Flip, метод наискорейшего спуска*

Источник финансирования: *статья подготовлена в рамках прикладных научных исследований СПбГУТ, регистрационный номер 1023031600087-9-2.2.4;2.2.5;2.2.6;1.2.1;2.2.3 в ЕГИСУ НИОКТР.*

Введение

Для приближения к потенциальной помехоустойчивости требуется применять помехоустойчивые коды с большой длиной блока (в десятки и сотни тысяч бит). Построить практические декодеры с такой диной блока до настоящего времени позволяют только коды с низкой плотностью проверок на четность (Low-Density Parity-Check Codes, LDPC), называемые также для краткости низкоплотностными [1, 2]. При этом из применяемых на практике декодеров наилучшие

Библиографическая ссылка на статью:
Копылов Д. А., Мендельсон М. А., Егоров В. А., Лютин В. И. Декодирование кодов LDPC с регулярной структурой проверочной матрицы способом Bit Flip // Информационные технологии и телекоммуникации. 2024. Т. 12. № 1. С. 40–56. DOI: 10.31854/2307-1303-2024-12-1-40-56. EDN: FMEVCV

Reference for citation:
Kopylov D., Mendelson M., Egorov V., Lyutin V. Decoding LDPC Codes with a Regular Check Matrix Structure Using the Bit Flip Method. *Telecom IT*. 2024. Vol. 12. Iss. 1. PP. 40–56. (in Russian). DOI: 10.31854/2307-1303-2024-12-1-40-56. EDN: FMEVCV

показатели по исправлению ошибок демонстрируют методы декодирования, основанные на алгоритме «Сумма – Произведение» (Sum-Product Algorithm, SPA), называемом также алгоритмом с распространением доверия (Belief Propagation, BP) [1, 3]. Алгоритм SPA требует применение мягкого решения, например, отношения правдоподобия для каждого принятого бита. Несколько более простой алгоритм Min-Sum («минимальная сумма») также работает с мягким решением [4].

Использование мягкого решения усложняет не только алгоритм работы демодулятора, но и алгоритм декодирования принятой последовательности, поэтому в случае жестких ограничений производительности оборудования оказывается необходимым предельное упрощение алгоритма реализации приемного устройства в целом.

Р. Галлагер первым предложил альтернативные алгоритмы декодирования LDPC, основанные на жестком решении [5], именуемые алгоритмами с инверсией бит (Bit Flip, BF) [1, 6–9]. Общий принцип их работы состоит в том, что на каждой последующей итерации вычисляются так называемые функции инверсии (Flip Functions, FF) и по вычисленным значениям FF один или несколько бит инвертируется (переворачивается). Отсюда и наименование алгоритма BF (Flip – переворачивать). Функции инверсии вычисляются для каждого бита в принятом блоке и представляют собой некоторую метрику надежности временно принятого решения для данного бита, основанную на вычисленных контрольных суммах, в которые данный бит входит. Алгоритм BF намного проще в реализации, хотя и уступает алгоритму SPA в помехоустойчивости. Для уменьшения проигрыша в помехоустойчивости было предложено много улучшений алгоритма BF, в некоторых из них варьируется метрика надежности бит (FF), в других – метод выбора инвертируемых бит. При этом, как правило, достигается уменьшение коэффициента ошибок за счет некоторого увеличения сложности алгоритма.

В работе [7] предложен класс алгоритмов BF с применением вместо двоичных контрольных сумм дробных весов, которые можно рассматривать как метрики надежности контрольных сумм. Эти метрики являются некоторой функцией мягких решений для принятых бит [7–9]. Для обозначения алгоритмов этого типа применяют аббревиатуру WBF (Weighted Bit Flip – инверсия бит со взвешиванием). В работе [10] предложен алгоритм декодирования кодов LDPC, который назван алгоритмом инверсии бит с применением градиентного спуска (Gradient Descent Bit-Flipping, GDBF). При реализации данного алгоритма FF вычисляются на основании градиента нелинейной целевой функции, которая эквивалентна логарифмической функции правдоподобия принятых бит при ограничениях на контрольные суммы. Кроме этого, известен также алгоритм WBF, в котором веса контрольных сумм изменяются в процессе декодирования в соответствии с нелинейной функцией от FF [1].

Среди алгоритмов BF одни могут производить инверсию только одного бита за итерацию, другие – сразу нескольких бит. Инвертирование только одного бита за итерацию замедляет процесс декодирования. Для увеличения скорости декодирования предложен целый ряд вариантов выбора для инвертирования за одну итерацию группы бит [10–16]. При инвертировании сразу нескольких бит скорость декодирования существенно повышается, однако в некоторых случаях

такой подход приводит к зацикливанию алгоритма, что ухудшает характеристики помехоустойчивости [1, 10].

Правило выбора одновременно инвертируемых бит может либо представлять собой простое сравнение с порогом, либо включать ряд шагов, которые принимают во внимание несколько метрик надежности бит. Наиболее сложные алгоритмы выбора инвертируемых бит основываются не только на вычисленных значениях FF, но и на дополнительной информации, извлекаемой из весов контрольных сумм [1].

Целью данной работы является нахождение упрощенного алгоритма декодирования кодов LDPC, имеющих проверочную матрицу с регулярной структурой (повторно-накопительные коды). При этом за основу приняты известные обобщенные алгоритмы декодирования кодов LDPC способом GDBF, имеющие наилучшие показатели помехоустойчивости для алгоритмов BF. Дополнительно учитываются ограничения на сложность реализации декодера, задаваемые параметрами широко используемых микросхем с программируемой логикой.

Способ декодирования GDBF

Рассмотрим подробнее общий принцип работы алгоритмов декодирования кодов LDPC, основанных на алгоритме BF с применением принципов GDBF.

Далее по тексту, в соответствии со сложившейся традицией, применяем следующие обозначения: n – длина кодового слова, k – число информационных бит, m – число проверочных бит, $n = k + m$. Параметры такого кода обозначаются (n, k).

На передаче формируется кодовый вектор LDPC:

$$\boldsymbol{C} = (c_0, c_1, \dots c_{n-1}), \quad \boldsymbol{C} \epsilon \widetilde{\boldsymbol{C}}, \tag{1}$$

где c_i – двоичный бит. В этой формуле i принимает значения от 0 до $n - 1$. Этот кодовый вектор $\boldsymbol{C}$ входит в состав множества допустимых кодовых слов $\widetilde{\boldsymbol{C}}$.

Как и для любых других линейных блочных кодов, параметры кода LDPC могут быть заданы проверочной матрицей $\boldsymbol{H}$. Для кода (n, k) проверочная матрица выглядит как прямоугольная матрица размером $n \times m$:

$$\boldsymbol{H} = \begin{bmatrix} h_{00} & h_{01} & \cdots h_{0(n-1)} \\ h_{10} & h_{11} & \cdots h_{1(n-1)} \\ h_{(m-1)0} & h_{(m-1)1} & \cdots h_{(m-1)(n-1)} \end{bmatrix}, \tag{2}$$

где h_{ij} – элемент проверочной матрицы, который соответствует i-му биту кодового блока и j-ой строке проверочной матрицы $\boldsymbol{H}$. В этой формуле i принимает значения от 0 до $n - 1$, а j принимает значения от 0 до $m - 1$. Каждый такой элемент проверочной матрицы может равняться 0 или 1. Каждая строка проверочной матрицы соответствует одному из проверочных уравнений. Для каждого кодового вектора выполняются все m проверочных уравнений:

$$\begin{cases} h_{00}c_0 \oplus h_{01}c_1 \oplus \cdots \oplus h_{0(n-1)}c_{(n-1)} = 0; \\ h_{10}c_0 \oplus h_{11}c_1 \oplus \cdots \oplus h_{1(n-1)}c_{(n-1)} = 0; \\ h_{(m-1)}c_0 \oplus h_{(m-1)1}c_1 \oplus \cdots \oplus h_{(m-1)(n-1)}c_{(n-1)} = 0. \end{cases} \quad (3)$$

Если при линейных преобразованиях проверочной матрицы $\boldsymbol{H}$ в одном из ее вариантов число единиц в каждой строке и в каждом столбце мало, то такой код относят к низкоплотностным. Можно сказать, что код LDPC является кодом с низкой плотностью единиц в проверочной матрице.

В канале на сигнал воздействуют помехи. Поэтому на выходе решающей схемы вырабатывается принятый вектор:

$$\boldsymbol{Y} = (y_0, y_1, \dots y_{n-1}), \quad (4)$$

где y_i – двоичный бит, соответствующий по положению исходному двоичному биту c_i. В этой формуле i принимает значения от 0 до $n-1$.

Обозначим вектор, получаемый на каждой следующей итерации процесса декодирования:

$$\boldsymbol{X} = (x_0, x_1, \dots x_{n-1}). \quad (5)$$

В принципе задача на приеме состоит в нахождении кодового вектора $\boldsymbol{C}$. Но реально достижимая задача – постараться найти такой вектор $\widehat{\boldsymbol{X}}$, который является одним из допустимых кодовых векторов $\widehat{\boldsymbol{X}} \epsilon \widetilde{\boldsymbol{C}}$, и при этом при известном принятом на выходе решающей схемы векторе $\boldsymbol{Y}$ вектор $\widehat{\boldsymbol{X}}$ должен быть максимально правдоподобным.

Тогда правило вычисления декодированного вектора $\widehat{\boldsymbol{X}}$ по критерию максимального правдоподобия может быть в общем виде представлено следующим образом:

$$\widehat{\boldsymbol{X}} = \arg\min_{\boldsymbol{X} \in \widetilde{\boldsymbol{C}}} \sum_{i=0}^{n-1} (x_i \oplus y_i), \quad (6)$$

где $x_i \oplus y_i$ – сумма по модулю 2 соответствующих бит вектора на выходе решающей схемы и декодированного вектора, а знак суммы обозначает простое арифметическое сложение. Другими словами, результат вычисления суммы – число бит, которые отличаются в векторах $\widehat{\boldsymbol{X}}$ и $\boldsymbol{Y}$. В частном случае, если ошибок в принятом векторе $\boldsymbol{Y}$ нет, то векторы $\widehat{\boldsymbol{X}}$ и $\boldsymbol{Y}$ совпадают, а сумма в формуле (6) принимает минимально возможное значение, равное нулю. Если ошибки в принятом векторе $\boldsymbol{Y}$ имеются, то следует отыскать такой вектор $\widehat{\boldsymbol{X}}$, который принадлежит множеству допустимых кодовых векторов, а число отличающихся бит в векторах $\widehat{\boldsymbol{X}}$ и $\boldsymbol{Y}$ минимально.

DOI: 10.31854/2307-1303-2024-12-1-40-56
URL: https://www.sut.ru/doci/nauka/1AEA/ITT/2024_1/40-56.pdf

В процессе декодирования линейных кодов широко применяется понятие синдрома – двоичного вектора $\boldsymbol{S}$ размерности m, состоящего из m элементов s_j, которые вычисляются по формулам, аналогичным проверочным уравнениям:

$$\begin{cases} s_0 = h_{00}x_0 \oplus h_{01}x_1 \oplus \cdots \oplus h_{0(n-1)}x_{(n-1)}; \\ s_1 = h_{10}c_0 \oplus h_{11}c_1 \oplus \cdots \oplus h_{1(n-1)}c_{(n-1)}; \\ s_{(m-1)} = h_{(m-1)}c_0 \oplus h_{(m-1)1}c_1 \oplus \cdots \oplus h_{(m-1)(n-1)}c_{(n-1)}; \end{cases} \quad (7)$$

в этой формуле j принимает значения от 0 до $m - 1$.

Для кодовых векторов синдром равен нулевому вектору, для всех остальных векторов синдром не равен нулевому вектору, т. е. часть проверочных уравнений не выполняется. Величина $W = \sum_{j=0}^{m-1} s_j$, равная арифметической сумме всех элементов синдрома, называется весом синдрома $\boldsymbol{S}$. По сути вес синдрома W равен числу проверочных уравнений, которые не выполняются для данного вектора $\boldsymbol{X}$. Для кодовых векторов вес синдрома равен нулю.

Один из вариантов целевой функции, применяемых в алгоритмах декодирования кодов LDPC, основанных на алгоритме BF с применением GDBF [10], можно представить следующим образом:

$$f(\boldsymbol{X}) = \sum_{i=0}^{n-1}(x_i \oplus y_i) + \alpha \sum_{j=0}^{m-1} s_j, \quad (8)$$

где α – постоянный коэффициент. Для данной целевой функции ищется минимум. Первое слагаемое при минимизации целевой функции $f(\boldsymbol{X})$ дает минимум числа отличий бит текущего значения декодированного вектора $\boldsymbol{X}$ от принятого (с ошибками) вектора $\boldsymbol{Y}$. Второе слагаемое минимизирует вес синдрома для этого вектора $\boldsymbol{X}$. Коэффициент α устанавливается достаточно большим для того, чтобы полученный в результате декодирования вектор $\widehat{\boldsymbol{X}}$ гарантированно входил в число кодовых слов, в соответствии с формулой (6).

В работе [10] подробно описаны конкретные алгоритмы декодирования GDBF. Поведение процесса декодирования GDBF имеет много общего с поведением алгоритмов оптимизации в случае нелинейной целевой функции, в которых используется метод наискорейшего спуска.

Для алгоритма GDBF с инвертированием только одного бита за итерацию поведение процесса декодирования до достижения локального минимума схематично представлено на рисунке 1а. Небольшой шаг приближения к локальному минимуму, вызванный инвертированием только одного бита за итерацию, приводит к большому числу итераций, и, как следствие, к низкой скорости декодирования. При этом осуществляется самое близкое приближение к локальному минимуму.

Рис. 1. Схема поведения процесса декодирования GDBF в зоне локального минимума: а) шаг – один бит; б) шаг – группа бит; в) переменный шаг

При применении алгоритма GDBF с инвертированием сразу группы бит осуществляется быстрое продвижение в направлении локального минимума, что в целом приводит к уменьшению числа итераций декодирования. Тем не менее, при фиксированных параметрах выбора группы инвертируемых бит при приближении к локальному минимуму очередная итерация может «проскочить» этот минимум, и, как следствие, происходит зацикливание алгоритма вблизи локального минимума. Примерное поведение процесса декодирования при применении алгоритма GDBF с инвертированием сразу группы бит и фиксированным правилом выбора группы инвертируемых бит поведение процесса декодирования до достижения локального минимума схематично представлено на рисунке 1б.

Альтернативный подход – применение алгоритма GDBF с переменным шагом. Во время первых итераций выбор группы инвертируемых бит происходит по тем же правилам, что и для алгоритма с инвертированием сразу группы бит и фиксированным правилом выбора группы инвертируемых бит. Далее, при приближении к локальному минимуму, правило выбора группы инвертируемых бит меняется в сторону уменьшения размера группы инвертируемых бит. Далее правило выбора инвертируемых бит переходит в правило выбора только одного инвертируемого бита. В результате число требуемых итераций лишь слегка превышает число итераций для алгоритма GDBF с инвертированием сразу группы бит и фиксированным правилом выбора группы инвертируемых бит. В то же время зацикливания при приближении к локальному минимуму не происходит, и точность достижения локального минимума соответствует точности, достигаемой при применении алгоритма с инвертированием только одного бита за итерацию. Поведение процесса декодирования при применении алгоритма GDBF с переменным шагом до достижения локального минимума схематично представлено на рисунке 1в.

Нелинейная целевая функция может иметь не один минимум, который соответствует искомому декодированному вектору, а несколько. Описанные выше алгоритмы GDBF не гарантируют, что будет найден именно глобальный мини-

мум. Нередко в процессе реального декодирования алгоритм GDBF «захватывает» локальный минимум, который не соответствует искомому декодированному вектору. Для выхода из этой ситуации можно воспользоваться результатами работы [10], где предлагается применить с этой целью процесс ESCAPE (выход из локального минимума). Попадание в локальный минимум детектируется тогда, когда инвертирование бит или прекращается, или зацикливается, в то время как вес синдрома не становится равным 0, т. е. промежуточный найденный вектор $\boldsymbol{X}$ не является кодовым. Тогда и применяется процедура ESCAPE, требующая внесения в алгоритм достаточно большого возмущения, для чего изменяется правило выбора группы инвертируемых бит не в сторону уменьшения числа инвертируемых бит (как это принято в алгоритме GDBF с переменным шагом), а в сторону их увеличения. Однократное применение инверсии сразу большого числа бит, как правило, приводит к удалению от минимума целевой функции, но нередко позволяет «выскочить» из локального минимума. Далее параметры выбора группы инвертируемых бит восстанавливаются, и продолжается поиск очередного локального минимума, который в итоге может стать глобальным, в результате чего искомый кодовый вектор $\widehat{\boldsymbol{X}}$ будет, в конце концов, найден. Поведение алгоритма GDBF для случая «захвата» локального минимума с применением процедуры ESCAPE схематично представлено на рисунке 2.

Разумеется, не каждая процедура ESCAPE приводит к выходу из данного локального минимума, и не каждая такая процедура приводит к попаданию в зону захвата глобального минимума. Тем не менее применение процедуры ESCAPE во многом способствует повышению помехоустойчивости алгоритмов декодирования GDBF [10].

Рис. 2. Процедура выхода из локального минимума

Коды LDPC с регулярной структурой

На практике стараются выбирать LDPC коды с регулярной структурой. Характеристики помехоустойчивости таких кодов при одинаковых параметрах n, k не отличаются от таких же характеристик для кодов LDPC со случайной структурой, но реализация декодеров заметно упрощается. Это относится к декодированию, основанному как на мягком, так и на жестком решении.

DOI: 10.31854/2307-1303-2024-12-1-40-56
URL: https://www.sut.ru/doci/nauka/1AEA/ITT/2024_1/40-56.pdf

Один из вариантов кода LDPC с регулярной структурой – повторно-накопительные коды (Repeat-Accumulate) [17]. Для них число единиц в m последних столбцах проверочной матрицы равно 2. Точнее, $m - 1$ столбцов имеют по две единицы, а самый последний – всего одну. Для примера рассмотрим короткий код (12, 3) с проверочной матрицей, представленной на рисунке 3, где желтым цветом выделены единицы в последних девяти столбцах проверочной матрицы.

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

Рис. 3. Пример проверочной матрицы повторно-накопительного кода (12, 3)

Регулярную структуру, которая, в частности, включает и Repeat-Accumulate (повторно-накопительную) часть, имеют LDPC коды, входящие в стандарты [18, 19]. Рассмотрим, например, конкретный код (n = 16 200, k = 14 400), имеющий обозначение «8/9».

В таблице 1 приведены начальные значения адресов бит аккумулятора для процесса кодирования LDPC.

Алгоритм формирования кодового вектора следующий. Первые k = 14 400 бит – информационная последовательность (любая последовательность бит); обозначим их (i_0, i_1, … i_{14399}). Оставшиеся m = 1800 бит – проверочные биты (p_0, p_1, … p_{1799}), которые формируются по следующему правилу: исходное состояние последовательности проверочных бит, вычисляемое в аккумуляторе, – нулевое:

$$p_0 = p_1 = \cdots = p_{1799} = 0. \quad (9)$$

Далее выполняется процесс накопления (аккумулирования), который выполняется для всех k = 14 400 информационных бит и представляет собой сложение по модулю 2 значений информационных бит с аккумулятором по выбранным адресам. Для самого первого бита i_0 накопления происходят для бит аккумулятора, адреса которых приведены в первой ячейке таблицы 1, а именно:

$$p_0 = p_0 \oplus i_0, \quad p_{1558} = p_{1558} \oplus i_0, \quad p_{712} = p_{712} \oplus i_0, \quad p_{805} = p_{805} \oplus i_0, \qquad (10)$$

где сложения выполняются по модулю 2.

Таблица 1 – Начальные значения адресов бит аккумулятора для групп кодовых бит по 360

№ ячейки	Начальные адреса	№ ячейки	Начальные адреса	№ ячейки	Начальные адреса	№ ячейки	Начальные адреса
1	0 1558 712 805	11	0 427 488	21	0 809 385	31	0 508 630
2	1 1450 873 1337	12	1 828 1124	22	1 367 151	32	1 421 1704
3	2 1741 1129 1184	13	2 874 1366	23	2 1323 202	33	2 284 898
4	3 294 806 1566	14	3 1500 835	24	3 960 318	34	3 392 577
5	4 482 605 923	15	4 1496 502	25	4 1451 1039	35	4 1155 556
6	0 926 1578	16	0 1006 1701	26	0 1098 1722	36	0 631 1000
7	1 777 1374	17	1 1155 97	27	1 1015 1428	37	1 732 1368
8	2 608 151	18	2 657 1403	28	2 1261 1564	38	2 1328 329
9	3 1195 210	19	3 1453 624	29	3 544 1190	39	3 1515 506
10	4 1484 692	20	4 429 1495	30	4 1472 1246	40	4 1104 1172

Для последующих 359 информационных бит (i1, i2, … i359) накопления производятся по формулам, аналогичным (10), но адреса бит аккумулятора вычисляются по формуле:

$$addr_j = (addr_0 + 5j)(mod\ 1800), \qquad (11)$$

где $addr_j$ – четыре адреса для информационного бита i_j, соответствующие четырем адресам $addr_0$, входящим в формулу (10) и определяемым первой ячейкой таблицы 1.

Например, для информационного бита i_1:

$$p_5 = p_5 \oplus i_1, \quad p_{1563} = p_{1563} \oplus i_1, \quad p_{717} = p_{717} \oplus i_1, \quad p_{810} = p_{810} \oplus i_1. \qquad (12)$$

Для информационного бита под номером 360 i_{360} накопление происходит аналогично правилу (10), но начальные адреса определяются из второй ячейки таблицы 1, а именно:

$$p_1 = p_1 \oplus i_{360},\ p_{1450} = p_{1450} \oplus i_{360},\ p_{873} = p_{873} \oplus i_{360},\ p_{1337} = p_{1337} \oplus i_{360}. \qquad (13)$$

Для последующих 359 информационных бит (i_{361}, i_{362}, … i_{719}) накопления производятся по аналогичным формулам, но адреса бит аккумулятора определяются по формуле (11).

Для следующих трех групп по 360 информационных бит начальные адреса бит аккумулятора определяются по следующим трем ячейкам таблицы 1.

Начиная с информационного бита с номером 1800 i_{1800} накопление производится для трех проверочных бит, в частности, для бита i_{1800}, для которого адреса бит аккумулятора задаются в шестой ячейке таблицы 1:

$$p_0 = p_0 \oplus i_{1800}, \quad p_{926} = p_{926} \oplus i_{1800}, \quad p_{1578} = p_{1578} \oplus i_{1800}. \tag{14}$$

Далее для каждой группы из 360 информационных бит накопления происходят аналогично, с использованием в качестве начальных адресов бит аккумулятора последующих ячеек таблицы 1.

После завершения процесса накопления для всех проверочных бит выполняются следующие операции, начиная с бита p_1:

$$p_j = p_j \oplus p_{j-1}, \; j = 1 \ldots 1799. \tag{15}$$

Полученная последовательность проверочных бит (p_0, p_1, … p_{1799}) добавляется справа к исходной информационной последовательности (i_0, i_1, … i_{14399}), формируя кодовое слово LDPC.

Декодирование кодов LDPC с регулярной структурой

Проверочная матрица кода LDPC «8/9», описанного выше, может быть представлена в следующем виде. Размер проверочной матрицы: 1800 строк на 16 200 столбцов. В первых 1800 столбцах имеется по четыре единицы. Например, в самом левом столбце под номером 0 четыре единицы находятся в строках, которые определяются номерами из первой ячейки таблицы 1 (0, 1558, 712, 805). В последующих 359 столбцах находятся по четыре единицы, номера строк для которых определяются так же, как и адреса по формуле (11). В следующих четырех группах по 360 столбцов также имеется по четыре единицы, адреса строк которых определяются с помощью следующих четырех ячеек таблицы 1. Начиная со столбца под номером 1800, в столбцах проверочной матрицы имеется по три единицы. Для всех следующих 35 групп по 360 столбцов начальные адреса строк проверочной матрицы определяются по значениям в следующих ячейках таблицы 1.

В предпоследних 1799 столбцах проверочной матрицы записаны по две единицы, а в последнем столбце – одна единица, аналогично проверочной матрице, приведенной на рисунке 3. Структура проверочной матрицы рассматриваемого кода показана на рисунке 4.

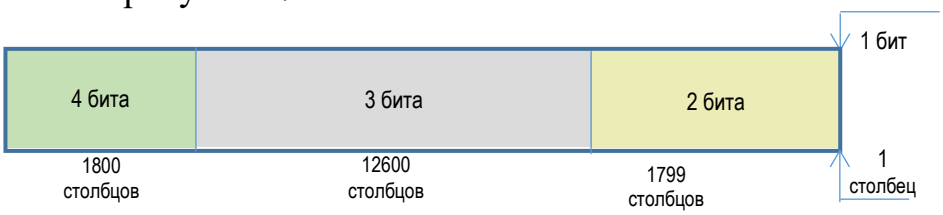

Рис. 4. Структура проверочной матрицы кода LDPC 8/9

DOI: 10.31854/2307-1303-2024-12-1-40-56
URL: https://www.sut.ru/doci/nauka/1AEA/ITT/2024_1/40-56.pdf

В данной работе предлагается следующий вариант вычисления функций инверсии бит (FF). *Для каждого из первых 1800 бит (0…1799)* возможно максимально четыре строки проверочной матрицы, для которых не выполняется проверка при вычислении синдрома. Поэтому по результату вычисления синдрома число не выполнившихся проверок может быть: 4 из 4, 3 из 4, 2 из 4, 1 из 4 или 0 из 4. Соответственно, для каждого из первых 1800 бит можно вычислить функцию FF_4, принимающую значения: 0, 1, 2, 3, 4. *Для каждого из следующих 12 600 бит (1800…14 399)* возможно максимально три строки проверочной матрицы, для которых не выполняется проверка при вычислении синдрома. Поэтому по результату вычисления синдрома число не выполнившихся проверок может быть: 3 из 3, 2 из 3, 1 из 3 или 0 из 3. Соответственно, для каждого из этих 12 600 бит можно вычислить функцию FF_3, принимающую значения: 0, 1, 2, 3. *Для каждого из следующих 1799 бит (14 400…16 198)* возможно максимально две строки проверочной матрицы, для которых не выполняется проверка при вычислении синдрома. Поэтому по результату вычисления синдрома число не выполнившихся проверок может быть: 2 из 2, 1 из 2 или 0 из 2. Соответственно, для каждого из этих 1799 бит можно вычислить функцию FF_2, принимающую значения: 0, 1, 2. *Для последнего бита (16 199)* возможна максимально одна строка проверочной матрицы, для которой не выполняется проверка при вычислении синдрома. Поэтому по результату вычисления синдрома число не выполнившихся проверок может быть: 1 из 1 или 0 из 1. Соответственно, для этого бита можно вычислить функцию FF_1, принимающую значения: 0, 1.

Наиболее правдоподобными для инвертирования являются биты, для которых FF_4 = 4, и далее по убывающей: FF_3 = 3, FF_2 = 2, FF_1 = 1, FF_4 = 3, FF_3 = 2. Поэтому при определении группы бит для инвертирования на начальных итерациях выбираются все биты с функциями инверсии: FF_4 = 4, FF_3 = 3, FF_2 = 2, FF_1 = 1.

На последующих итерациях выбираются группы бит меньшего размера. Сначала выбираются те биты, для которых FF_4 = 4; когда такие биты заканчиваются, выбираются те биты, для которых FF_3 = 3, и далее таким же образом FF_2 = 2, FF_1 = 1. При этом после выполнения каждой итерации, инверсии выбранных бит и нового вычисления синдрома, могут снова появиться биты, для которых выполняются некоторые из предыдущих равенств. Далее по такому же принципу выбираются биты для инвертирования по одному. Когда после некоторого числа итераций процесс зацикливается, требуется произвести процедуру выхода из локального минимума (ESCAPE), которая может задействовать инвертирование группы бит, включающих не только те, для которых FF_4 = 3, FF_3 = 2, но также и FF_4 = 2, а также FF_2 = 1. При необходимости процедуру ESCAPE можно выполнить еще раз, но выбрать для инвертирования не все такие биты, а какую-то часть из них. В следующий раз можно выбрать другую часть бит из такого списка.

Приведенный алгоритм для декодирования конкретного кода LDPC «8/9» можно распространить и на другие варианты кодов LDPC с систематической структурой, которые стандартизированы в [18, 19]. В общем виде предложенный

алгоритм декодирования кода LDPC с жестким решением можно сформулировать следующим образом. До начала итераций исходное значение текущего вектора $\boldsymbol{X}$ выбирается равным принятому вектору $\boldsymbol{Y}$ на выходе решающей схемы с жестким решением. Текущее число итераций устанавливается в начальное значение, равное единице. Каждая итерация состоит из представленных ниже этапов.

Этап 1. Вычисление синдрома для текущего вектора $\boldsymbol{X}$.

Этап 2. Если вес полученного синдрома $W = 0$, то процесс декодирования заканчивается успешно. Декодированный вектор данного блока найден.

Этап 3. Если вес полученного синдрома не равен 0, то увеличивается текущее число итераций. Если текущее число итераций равно максимально допустимому, то процесс декодирования для данного блока завершается. Декодированный вектор данного блока не найден.

Этап 4. Если текущее число итераций меньше максимально допустимого, то процесс декодирования продолжается вычислением функций инверсии бит (FF_4, FF_3, FF_2, FF_1).

Этап 5. Выбирается группа бит для инвертирования. Этот выбор зависит от результата вычисления функций инверсии бит. Эта группа может быть большого или среднего размера, или включать только один бит, как описано ранее. В случае процедуры ESCAPE группа бит для инвертирования может быть максимально широкой.

Этап 6. Производится инвертирование выбранной группы бит. Формируется новое значение для текущего вектора $\boldsymbol{X}$. Переход к этапу 1.

При практической реализации имеет смысл применять распараллеливание вычислений. Например, для вычисления синдрома можно предложить одновременное вычисление нескольких частичных сумм по модулю 2 для нескольких частей групп бит кодового вектора с последующим суммированием по модулю 2 этих частичных сумм. Это требует увеличения объема задействованной памяти, но существенно уменьшает время выполнения каждой итерации. В результате увеличивается число разрешенных итераций, что приводит к улучшению характеристик помехоустойчивости. Аналогичные принципы распараллеливания можно применять и для вычисления функций инверсии бит, и для инвертирования выбранной группы бит.

Кроме того, можно учесть, что число бит, для которых требуется производить инвертирование, может зависеть от выбора группы инвертируемых бит. Например, если группа инвертируемых бит содержит только те биты, для которых выполняются равенства FF_4 = 4, FF_4 = 3, то общее число бит для рассмотрения – это только первые 1800. Аналогично, если группа инвертируемых бит содержит только те биты, для которых выполняются равенства FF_2 = 2, FF_1 = 1, то общее число бит для рассмотрения – это только последние 1800. Значит, в этих случаях время инвертирования бит может быть сокращено. Если на каком-то этапе алгоритма выполняется инвертирование только одного конкретного бита, то время инвертирования вообще ничтожно. Это означает, что длительность выполнения итерации не является фиксированной. Поэтому более

логично ограничивать не количество итераций, а общее время выполнения итеративного процесса декодирования блока. Это может в ряде случаев привести к увеличению числа реально выполненных итераций, а, следовательно, к улучшению характеристик помехоустойчивости. Имеется дополнительная возможность увеличить число итераций (время декодирования) для конкретного блока, если допустить некоторую задержку выполнения декодирования. Допустим, до начала процесса декодирования в памяти декодера накоплено N принятых блоков (N = 2…5). Тогда, если первый из блоков будет декодирован за время меньшее максимально допустимого, может высвободиться дополнительное время для декодирования одного из следующих блоков, что тоже может привести к улучшению характеристик помехоустойчивости.

Результаты моделирования для кодов LDPC, стандартизованных в [18], для длины блока n = 64 800 и относительных скоростей 3/4, 4/5, 5/6, 8/9, приведены на рисунке 5, где показаны графики зависимостей коэффициента ошибок (Kош) от отношения сигнал/шум (ОСШ) для методов декодирования BP, стандартного BF и рассмотренного GDBF, используемого для кодов LDPC, имеющих проверочную матрицу с регулярной структурой

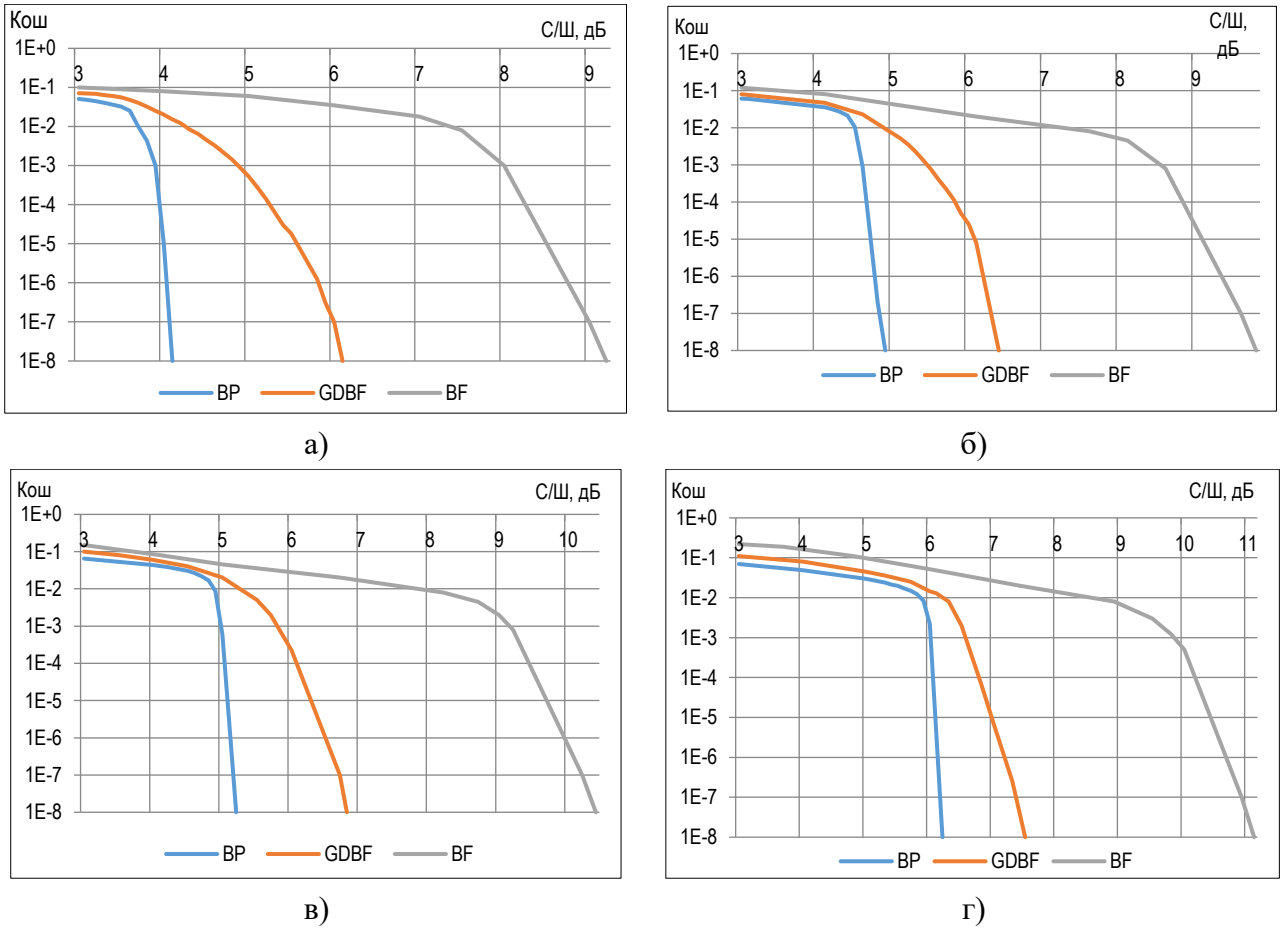

Рис. 5. Результаты моделирования декодирования кода LDPC с относительной скоростью: 3/4 (а), 4/5 (б), 5/6 (в) и 8/9 (г)

DOI: 10.31854/2307-1303-2024-12-1-40-56
URL: https://www.sut.ru/doci/nauka/1AEA/ITT/2024_1/40-56.pdf

На всех графиках предполагается, что в канале применяется метод модуляции QPSK (*аббр. от англ*. Quadrature Phase Shift Keying, квадратурная фазовая манипуляция), и максимальное разрешенное число итераций равно 100.

Очевидно, что с использованием предлагаемого алгоритма выигрыш по сравнению со стандартным алгоритмом BF (жесткое решение) составляет 3-4 дБ, а проигрыш алгоритму с мягким решением уменьшился до 1,5-2 дБ.

Выводы

Поиск наиболее помехоустойчивых алгоритмов декодирования с жестким решением целесообразен с точки зрения упрощения реализации как демодулятора, так и декодера.

В статье предлагается алгоритм GDBF с применением процедуры выхода из локального минимума для конкретного кода LDPC, проверочная матрица которого имеет регулярную структуру.

Показано, что при использовании предлагаемого алгоритма выигрыш по сравнению со стандартным алгоритмом BF (жесткое решение) составляет 3-4 дБ, а проигрыш алгоритму с мягким решением (BP) уменьшился до 1,5-2 дБ.

Дальнейшие исследования могут быть связаны с применением принципа градиентного спуска с процедурой выхода из локального минимума для алгоритмов декодирования, основанных на мягком решении.

Литература

1. Chang Tofar C.-Y., Su Yu T. Dynamic Weighted Bit-Flipping Decoding Algorithms for LDPC Codes // IEEE Transactions on Communications. 2015. Vol. 63. Iss. 11. DOI: 10.1109/TCOMM.2015.2469780

2. MacKay D. J. C. Good Error-Correcting Codes Based on Very Sparse Matrices // IEEE Transactions on Information Theory. 1999. Vol. 45. Iss. 2. PP. 399–431. DOI: 10.1109/18.748992

3. Devrari A., Kumar A., Chauhan H., Kumar A. Design and FPGA Implementation of LDPC Decoder Chip for Communication System using VHDL // International Journal of Recent Technology and Engineering (IJRTE). 2019. Vol. 8, Iss. 2. DOI: 10.35940/ijrte.A2203.078219

4. Islam M. R., Shafiullah D. S., Faisal M. M. A., Rahman I. Optimized Min-Sum Decoding Algorithm for Low Density Parity Check Codes // International Journal of Advanced Computer Science and Applications. 2011. Vol. 2. No. 12. PP. 168–174. DOI: 10.14569/IJASA.2011.021225

5. Gallager R. G. Low-Density Parity-Check Codes. Cambridge, MA: MIT Press, 1963. 90 p.

6. Kou Y., Lin S., Fossorier M. P. C. Low-Density Parity-Check Codes Based on Finite Geometries: A Rediscovery and New Results // IEEE Transactions on Information Theory. 2001. Vol. 47. Iss. 7. PP. 2711−2736. DOI: 10.1109/18.959255

7. Zhang J., Fossorier M. P. C. A Modified Weighted Bit-Flipping Decoding of Low-Density Parity-Check Codes // IEEE Communications Letters. 2004. Vol. 8. Iss. 3. PP. 165–167. DOI: 10.1109/LCOMM.2004.825737

8. Jiang M., Zhao C., Shi Z., Chen Y. An Improvement on the Modified Weighted Bit Flipping Decoding Algorithm for LDPC Codes // IEEE Communications Letters. 2005. Vol. 9. Iss. 9. PP. 814–816. DOI: 10.1109/LCOMM.2005.1506712

9. Hanzo L., Guo F. Reliability Ratio Based Weighted Bit-Flipping Decoding for Low-Density Parity-Check Codes // IEEE Electronics Letters. 2004. Vol. 40. No. 21. PP. 1356–1358. DOI: 10.1049/el:20046400

10. Wadayama T. Nakamura K., Yagita M., Funahashi Y., Usami Sh. et al. Gradient Descent Bit Flipping Algorithms for Decoding LDPC Codes // IEEE Transactions on Communications. 2010. Vol. 58. Iss. 6. PP. 1610–1614. DOI: 10.1109/TCOMM.2010.06.090046

11. Wu X., Zhao C., You X. Parallel weighted bit-flipping decoding // IEEE Communications Letters. 2007. Vol. 11. Iss. 8. PP. 671–673. DOI: 10.1109/LCOMM.2007.070269

12. Li G., Feng G. Improved Parallel Weighted Bit-Flipping Decoding Algorithm for LDPC Codes // IET Communications. 2009. Vol. 3. Iss. 1. PP. 91–99. DOI: 10.1049/iet-com:20070632

13. Cho J., Sung W. Adaptive Threshold Technique for Bit-Flipping Decoding of Low-Density Parity-Check Codes // IEEE Communications Letters. 2010. Vol. 14. Iss. 9. PP. 857–859. DOI: 10.1109/LCOMM.2010.072310.100599

14. Haga R., Usami S. Multi-Bit Flip Type Gradient Descent Bit Flipping Decoding Using No Thresholds // Proceedings of 2012 International Symposium on Information Theory and its Applications (Honolulu, 28–31 Oct. 2012). PP. 6–10.

15. Ismail M., Ahmed I., Coon J. P., Armour S., Koçak T. et al. Low Latency Low Power Bit Flipping Algorithms for LDPC Decoding // Proceedings of 21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (Istanbul, 26–30 Sept. 2010). PP. 278–282. DOI: 10.1109/PIMRC.2010.5671820

16. Chen T.-Ch. Adaptive-Weighted Multibit-Flipping Decoding of Low-Density Parity-Check Codes Based on Ordered Statistics // IET Communications. 2013. Vol. 7. Iss. 14. PP. 1517–1521. DOI: 10.1049/iet-com.2013.0052

17. Johnson S. J. Introducing Low-Density Parity-Check Codes. Research Gate, May 2010. URL: https://www.researchgate.net/publication/228977165_Introducing_Low-Density_Parity-Check_Codes

18. ETSI EN 302 307-1 v.1.4.1 (2014-07). Digital Video Broadcasting (DVB); Second Generation Framing Structure, Channel Coding and Modulation Systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications. Part 1: DVB-S2.

19. ETSI EN 302 307-2 v.1.3.1 (2031-07). Digital Video Broadcasting (DVB); Second Generation Framing Structure, Channel Coding and Modulation Systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications. Part 2: DVB-S2 Extensions (DVB-S2X).

Статья поступила 19 сентября 2024 г.
Одобрена после рецензирования 28 сентября 2024 г.
Принята к публикации 22 октября 2024 г.

Информация об авторах

Копылов Дмитрий Андреевич – кандидат технических наук, начальник научно-исследовательской лаборатории систем передачи информации Санкт-Петербургского государственного университета телекоммуникаций им. проф. М. А. Бонч-Бруевича. E-mail: kopylov.da@sut.ru

Мендельсон Марк Александрович – кандидат технических наук, доцент, ведущий научный сотрудник научно-исследовательской лаборатории систем передачи информации Санкт-Петербургского государственного университета телекоммуникаций им. проф. М. А. Бонч-Бруевича.
E-mail: mendelson.ma@sut.ru

Егоров Вадим Анатольевич – кандидат технических наук, ведущий научный сотрудник научно-исследовательской лаборатории систем передачи информации Санкт-Петербургского государственного университета телекоммуникаций им. проф. М. А. Бонч-Бруевича. E-mail: egorov.va@sut.ru

Лютин Владимир Иванович – кандидат технических наук, доцент, старший научный сотрудник Научно-исследовательского испытательного института (радиоэлектронной борьбы) Военного учебно-научного центра Военно-воздушных сил «Военно-воздушная академия им. проф. Н. Е. Жуковского и Ю. А. Гагарина» (НИИИ (РЭБ) ВУНЦ ВВС «ВВА»). E-mail: lyutin_v_i@mail.ru

Decoding LDPC Codes with a Regular Check Matrix Structure Using the Bit Flip Method

D. Kopylov[1], **M. Mendelson**[1], **V. Egorov**[1], **V. Lyutin**[2]

[1]The Bonch-Bruevich Saint Petersburg State University of Telecommunications, St. Petersburg, 193232, Russian Federation
[2]Air Force Academy named after Professor N. E. Zhukovsky and Y. A. Gagarin, Voronej, 394064, Russian Federation

Problem statement: *low-density LDPC codes have become very popular in modern communication systems, which make it possible to implement decoders for them in practice. In conditions of severe limitations of equipment performance, the task of searching and researching the characteristics of decoding algorithms for these codes is urgent.* ***The aim of the work*** *is to find a simplified algorithm for decoding LDPC codes and to study the noise immunity of the proposed algorithm.* ***The methods used:*** *to solve this problem computer simulation is used to compare the noise immunity of known decoding algorithms and the proposed algorithm.* ***Scientific novelty:*** *the noise immunity characteristics of the proposed Bit Flip algorithm for decoding LDPC code with a regular structure of the check matrix using gradient descent and the procedure for escaping the local minimum are investigated.* ***Result:*** *a specific variant of the Bit Flip algorithm for LDPC code with a regular structure check matrix using gradient descent and a procedure for escaping the local minimum is proposed. The results of computer simulation are presented.* ***Theoretical significance:*** *the characteristics of noise immunity for the proposed variant of the LDPC code decoding algorithm with a regular structure of the check matrix are investigated.* ***Practical significance:*** *when using the proposed LDPC code decoding algorithm in the receiver, it is possible to calculate the parameters of the communication system based on the available dependencies of the error coefficient on the signal-to-noise ratio.*

Key words: *noise-resistant codes, LDPC code, Check Matrix, Bit Flip, Gradient descent method*

Funding: *the article was prepared within the framework of applied scientific research SPbSUT, Reg. No. 1023031600087-9-2.2.4;2.2.5;2.2.6;1.2.1;2.2.3 in the information system (https://www.rosrid.ru/information).*

Information about Authors

Kopylov Dmitriy – Candidate of Science, Head of the Laboratory (The Bonch-Bruevich Saint Petersburg State University of Telecommunications).
E-mail: kopylov.da@sut.ru

Mendelson Mark – Candidate of Science, Leading Scientific Researcher (The Bonch-Bruevich Saint Petersburg State University of Telecommunications).
E-mail: mendelson.ma@sut.ru

Egorov Vadim – Candidate of Science, Leading Scientific Researcher (The Bonch-Bruevich Saint Petersburg State University of Telecommunications).
E-mail: egorov.va@sut.ru

Lyutin Vladimir – Candidate of Science, Senior Scientific Researcher (Scientific Research Testing Institute (Electronic Warfare) of The Military Educational and Scientific Center of the Air Force «Air Force Academy named after Professor N. E. Zhukovsky and Y. A. Gagarin»). E-mail: lyutin_v_i@mail.ru