Задания для самостоятельной работы по помехоустойчивому кодированию

П.В. Трифонов

7 ноября 2022 г.

1 Список заданий

Задание	ссылка	баллы
Алгоритм Витерби мягкого декодирования линейных блоковых кодов	[2]	15
Алгоритм Витерби мягкого декодирования сверточных кодов	[2]	15
Алгоритм рекурсивного мягкого декодирования по решеткам линейных блоковых кодов	[8]	50
Алгоритм box&match мягкого декодирования линейных блоковых кодов	[10]	50
Алгоритм Берлекэмпа-Месси декодирования двоичных кодов БЧХ	[2]	15
Алгоритм Берлекэмпа-Месси декодирования кодов Рида-Соломона	[2]	15
Алгоритм Берлекэмпа-Месси декодирования двоичных кодов Гоппы	[2, 1]	17
Алгоритм Сугиямы (Евклида) декодирования двоичных кодов БЧХ	[1]	15
Алгоритм Сугиямы (Евклида) декодирования кодов Рида-Соломона	[1]	15
Алгоритм Сугиямы (Евклида) декодирования двоичных кодов Гоппы	[1]	17
Алгоритм распространения доверия декодирования LDPC кодов	[2]	30
Итеративное декодирование турбо-кодов	[2]	30
Алгоритм Гурусвами-Судана списочного декодирования кодов Рида-Соломона	[4, 5, 6]	40
Алгоритм Ву списочного декодирования кодов Рида-Соломона	[4, 5, 6, 11]	50
Списочное декодирование полярных кодов	[7]	40
Алгоритм последовательного исключения декодирования полярных кодов	[3, 7]	30
Последовательное декодирование полярных кодов	[7, 9]	50

Допускается реализация нескольких алгоритмов из вышеперечисленного списка. Не допускается реализация нескольких одноименных алгоритмов декодирования или различных алгоритмов декодирования одного и того же класса кодов. По усмотрению преподавателя может производиться собеседование, в ходе которого студент должен объяснить структуру программы и положенные в ее основу алгоритмы и теоретические принципы. В случае неверных ответов оценка может быть снижена.

Во всех случаях исходные данные для работы программы задаются в текстовом файле input.txt, а результат должен быть сохранен в файле output.txt.

Файл input.txt содержит описание кода, специфичное для задачи, а также одну или несколько инструкций, соответствующих следующим режимам работы (если ниже не указано иное):

- Кодирование указанного информационного вектора в заданном коде. Синтаксис команды:

  ```
  Encode  InfVector
  ```

  Пример:

```
Encode  0 0 1 0 1 0
```

После обработки команды Encode необходимо записать в выходной файл строку, соответствующую полученному кодовому слову. Отдельные символы должны разделяться пробелами.

- Decode — декодирование указанного вектора в заданном коде. В случае жесткого декодирования оно должно производиться в метрике Хемминга, при этом элементы указанного вектора должен рассматриваться как принадлежащие соответствующему конечному полю. В случае мягкого декодирования элементами входного вектора являются логарифмические отношения правдоподобия $L_i = \ln \frac{P(c_i=0|y_i)}{P(c_i=1|y_i)}$. В выходном файле должно быть представлено восстановленное кодовое слово, список кодовых слов, разделенных запятыми, или сообщение ERROR в случае невозможности декодирования.

  ```
  Decode  NoisyVector
  ```

  Пример:

  ```
  Decode  0 0 1 0 1 0
  ```

- Simulate — моделирование процесса передачи случайных данных в канале с указанным уровнем помех (смысл данного параметра указан в соответствующем задании). Должно быть выполнено NumOfIterations итераций генерация данных-кодирование-зашумление-декодирование или должно быть зарегистрировано MaxErrors ошибок декодирования. В выходном файле должна быть приведена полученная частота ошибок декодирования на кодовое слово.

  ```
  Simulate  NoiseLevel NumOfIterations MaxErrors
  ```

  Пример:

  ```
  Simulate  3.5 100000 100
  ```

Отдельными заданиями могут быть предусмотрены дополнительные команды и требования к выходному файлу. Результаты обработки различных инструкций в выходном файле должны разделяться символом перевода строки. В случае невозможности распознавания содержимого входного файла в выходной файл следует записать строку FAILURE. На выполнение каждой инструкции выделяется 1 минута.

2 Алгоритм Витерби мягкого декодирования линейных блоковых кодов

Необходимо реализовать кодер линейного блокового кода и его декодер по максимуму правдоподобия на основе алгоритма Витерби. Исходными данными являются длина n, размерность k и порождающая матрица G двоичного линейного блокового кода. Таким образом, файл input.txt должен начинаться следующим образом:

```
n k
G
```

Первой строкой в файле output.txt должна быть последовательность $|V_i|, 0 \leq i \leq n$, где $|V_i|$ — число узлов в решетке на ярусе i. Далее должны быть приведены результаты выполнения команд. Моделирование следует производить для случая канала с двоичной амплитудно-импульсной модуляцией и аддитивным белым гауссовским шумом. Под уровнем шума следует понимать отношение сигнал/шум на бит, выраженное в децибелах.

Пример файла input.txt:

```
8 4
1 1 1 1 1 1 1 1
1 1 1 1 0 0 0 0
1 1 0 0 1 1 0 0
1 0 1 0 1 0 1 0
Encode 1 0 0 0
Decode -1.0 1.0 1 1 1 1 1 1.5
Decode -10 1 1 1 1 1 1 1
Simulate 3 100000 100
Simulate 4 100000 100
```

Пример файла output.txt:

```
1 2 4 8 4 8 4 2 1
1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 0
2.56E-2
9.31E-3
```

3 Алгоритм Витерби мягкого декодирования сверточных кодов

Необходимо реализовать несистематический кодер сверточного кода со скоростью 1/2 и его декодер на основе алгоритма Витерби. После обработки заданной информационной последовательности кодер должен быть переведен в нулевое состояние путем принудительной подачи на его вход K нулевых битов, где K — длина кодового ограничения. Моделирование следует производить для случая канала с двоичной амплитудно-импульсной модуляцией и аддитивным белым гауссовским шумом. Под уровнем шума следует понимать отношение сигнал/шум на бит, выраженное в децибелах.

Первая строка в файле input.txt имеет вид

```
G0 G1 k
```

Здесь $G0, G1$ — порождающие многочлены кода, представленные в виде битовых масок в восьмеричной системе счисления, k — число кодируемых информационных символов (в десятичной системе счисления). Первая строка выходного файла должна содержать минимальное расстояние полученного блокового кода. Далее в выходном файле должны быть приведены результаты выполнения команд. Моделирование следует производить для случая канала с двоичной амплитудно-импульсной модуляцией и аддитивным белым гауссовским шумом. Под уровнем шума следует понимать отношение сигнал/шум на бит, выраженное в децибелах. Пример файла input.txt:

```
023 037 6
Encode 1 0 0 1 0 1
Decode -1 1.2 -1 -1 1 -1 -1 1 1 1 -1 1 -1 1 -1 1 1 -1 -1 -1
Simulate 4 100000 100
```

Пример файла output.txt:

```
6
1 1 1 1 0 1 1 0 0 0 1 0 1 0 1 0 0 1 1 1
1 1 1 1 0 1 1 0 0 0 1 0 1 0 1 0 0 1 1 1
1.02E-2
```

4 Алгоритм рекурсивного мягкого декодирования по решеткам линейных блоковых кодов

Необходимо реализовать кодер линейного блокового кода и его декодер на основе алгоритма Фудзивары-Ямамото-Касами-Линя (включая процедуру поиска оптимального секционирования). Формат входного файла совпадает с описанным в п. 2. В первой строке выходного файла должно быть указано число операций сложения и сравнения, выполняемых декодером. В последующих строках должны быть приведены результаты выполнения команд. Моделирование следует производить для случая канала с двоичной амплитудно-импульсной модуляцией и аддитивным белым гауссовским шумом. Под уровнем шума следует понимать отношение сигнал/шум на бит, выраженное в децибелах. Пример файла input.txt:

```
8 4
1 1 1 1 1 1 1 1
1 1 1 1 0 0 0 0
1 1 0 0 1 1 0 0
1 0 1 0 1 0 1 0
Encode 1 0 0 0
Decode -1.0 1.0 1 1 1 1 1 1.5
Decode -10 1 1 1 1 1 1 1
Simulate 3 100000 100
Simulate 4 100000 100
```

Пример файла output.txt:

```
23
1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 0
2.56E-2
9.31E-3
```

5 Алгоритм box&match мягкого декодирования линейных блоковых кодов

Необходимо реализовать кодер линейного блокового кода и его декодер на основе алгоритма box&match. Исходными данными являются длина n, размерность k и порождающая матрица G двоичного линейного блокового кода, а также параметры алгоритма "порядок переработки"(reprocessing order) t и длина контрольной полосы (control band) s. Таким образом, файл input.txt должен начинаться следующим образом:

```
n k
G
t s
```

В выходном файле должны быть приведены результаты выполнения команд. Пример файла input.txt:

```
32 16
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0
```

```
1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0
1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 5
Encode 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Decode -0.5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Simulate 3 100000 100
```

Пример файла output.txt

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.7E-3
```

6 Алгоритм Берлекэмпа-Месси декодирования двоичных кодов БЧХ

Необходимо реализовать процедуру построения двоичного примитивного кода БЧХ в узком смысле, его систематическое кодирование и декодирование в метрике Хемминга с помощью алгоритма Берлекэмпа-Месси. Файл input.txt должен содержать в первой строке длину кода $n = 2^m - 1$, примитивный многочлен поля $GF(2^m)$ (представленный в виде битовой маски в десятичной систем счисления) и конструктивное расстояние. Далее должны быть представлены команды. Под уровнем шума следует понимать вероятность ошибки на бит в двоичном симметричном канале. В первой строке выходного файла должна быть приведена размерность полученного кода k. На следующей строке должен быть приведен порождающий многочлен кода $g(x) = \sum_{i=0}^{n-k} g_i x^i$ в виде последовательности его коэффициентов $g_0, g_1, \ldots, g_{n-k}$. Далее должны быть представлены результаты выполнения команд. Пример файла input.txt:

```
7 11 3
Encode 1 0 0 0
Decode 0 1 0 1 0 0 0
Simulate 0.1 100000 100
```

Пример файла output.txt:

```
4
1 1 0 1
1 1 0 1 0 0 0
1 1 0 1 0 0 0
0.15
```

7 Алгоритм Сугиямы (Евклида) декодирования двоичных кодов БЧХ

Необходимо реализовать процедуру построения двоичного примитивного кода БЧХ в узком смысле, его систематическое кодирование и декодирование в метрике Хемминга с помощью расширенного алгоритма Евклида. Требования к входному и выходному файлу совпадают с приведенными в п. 6.

8 Алгоритм Берлекэмпа-Месси декодирования двоичных кодов Гоппы

Необходимо реализовать процедуру построения проверочной матрицы двоичного кода Гоппы длины $n = 2^m$, его систематическое кодирование и декодирование в метрике Хемминга. Для решения ключевого уравнения следует использовать алгоритм Берлекэмпа-Месси. Под уровнем шума следует понимать вероятность ошибки на бит в двоичном симметричном канале. В качестве локаторов кода при длиной 2^m следует рассматривать все элементы конечного поля в последовательности $0, \alpha^0, \alpha^1, \alpha^2, \alpha^3, \ldots, \alpha^{2^m-2}$. Для кодов длиной $2^m - 1$ следует исключить нулевой локатор. Гарантируется, что многочлен Гоппы свободен от квадратов и не имеет корней в $GF(2^m)$. В первой строке файла содержится длина кода 2^m или $2^m - 1$, примитивный многочлен поля $GF(2^m)$ (представленный в виде битовой маски в десятичной систем счисления) и степень многочлена Гоппы. Во второй строке выписаны коэффициенты многочлена Гоппы начиная с нулевого в виде чисел в десятичной системе счисления, двоичное представление которых соответствует их разложению по стандартному базису $GF(2^m)$. Далее следуют команды. В первой строке выходного файла должна быть выписана размерность кода. При построении порождающей матрицы кода в систематическом виде следует обеспечить размещение единичной матрицы в позициях с наименьшими возможными номерами.

Пример файла input.txt:

```
32 37 2
1 1 1
Encode 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Decode 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1
```

Пример файла output.txt:

```
22
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1
```

9 Алгоритм Сугиямы (Евклида) декодирования двоичных кодов Гоппы

Необходимо реализовать процедуру построения проверочной матрицы двоичного кода Гоппы длины $n = 2^m$, его систематическое кодирование и декодирование в метрике Хемминга. Для решения ключевого уравнения следует использовать алгоритм Сугиямы (расширенный алгоритм Евклида). Требования аналогичны приведенным в п. 8.

10 Алгоритм Берлекэмпа-Месси декодирования кодов Рида-Соломона

Необходимо реализовать процедуру построения примитивного кода Рида-Соломона в узком смысле над $GF(2^m)$, его систематическое кодирование и декодирование в метрике Хемминга с помощью алгоритма Берлекэмпа-Месси. Файл input.txt должен содержать в первой строке длину кода $n = 2^m - 1$, примитивный многочлен поля $GF(2^m)$ (представленный в виде битовой маски в десятичной систем счисления) и конструктивное расстояние. Далее должны быть представлены команды. Под уровнем шума следует понимать вероятность ошибки на символ в 2^m-ичном симметричном канале. Ненулевые символы вектора ошибки должны принимать значения из $GF(2^m) \setminus \{0\}$ с одинаковой вероятностью. В первой строке выходного файла должна быть приведена размерность полученного кода k. Далее должен быть

приведен порождающий многочлен кода $g(x) = \sum_{i=0}^{n-k} g_i x^i$ в виде последовательности его коэффициентов $g_0, g_1, \ldots, g_{n-k}$. Каждый коэффициент должен быть представлен в виде числа в десятичной системе счисления, двоичное представление которого соответствует его разложению по стандартному базису $GF(2^m)$. Далее должны быть представлены результаты выполнения команд. Пример файла input.txt:

```
7 11 3
Encode 1 0 0 0 0
Decode 3 6 1 0 0 0 1
Simulate 0.1 100000 100
```

Пример файла output.txt:

```
5
3 6 1
3 6 1 0 0 0 0
3 6 1 0 0 0 0
0.15
```

11 Алгоритм Сугиямы (Евклида) декодирования двоичных кодов Рида-Соломона

Необходимо реализовать процедуру построения примитивного кода Рида-Соломона в узком смысле над $GF(2^m)$, его систематическое кодирование и декодирование в метрике Хемминга с помощью расширенного алгоритма Евклида. Требования к входному и выходному файлу совпадают с приведенными в п. 10

12 Алгоритм Гурусвами-Судана списочного декодирования кодов Рида-Соломона

Необходимо реализовать процедуру построения примитивного кода Рида-Соломона в узком смысле над $GF(2^m)$, его систематическое кодирование и списочное декодирование в метрике Хемминга с радиусом декодирования до $\lfloor n - \sqrt{n(k-1)} \rfloor$ с использованием алгоритма Гурусвами-Судана. При моделировании следует считать ошибкой событие отсутствия в полученном списке переданного кодового слова. Требования к входному и выходному файлам аналогичны приведенными в п. 10. Если при выполнении команды Decode получается несколько кодовых слов, они должны быть выписаны в файле output.txt через запятую в порядке лексикографического возрастания как целочисленные вектора.

13 Алгоритм By списочного декодирования кодов Рида-Соломона

Необходимо реализовать процедуру построения примитивного кода Рида-Соломона в узком смысле над $GF(2^m)$, его систематическое кодирование и списочное декодирование в метрике Хемминга с радиусом декодирования до $\lfloor n - \sqrt{n(k-1)} \rfloor$ с использованием алгоритма By. Требования к входному и выходному файлам аналогичны приведенным в п. 12.

14 Алгоритм распространения доверия декодирования LDPC кодов

Необходимо реализовать систематическое кодирование и декодирование методом распространения доверия (в вероятностной области) кода с малой плотностью проверок на четность. При систематическом кодировании информационные символы следует располагать на позициях с наименьшими возможными номерами. Файл input.txt на первой строке содержит длину кода, число строк в проверочной матрице и максимальное число итераций в алгоритме распространения доверия. Далее приведено описание строк проверочной матрицы. В начале каждой строки указан ее вес, после чего выписан список номеров столбцов, содержащих единицы (нумерация с 0). Затем приведены команды. Моделирование следует производить для случая канала с двоичной амплитудно-импульсной модуляцией и аддитивным белым гауссовским шумом. Под уровнем шума следует понимать отношение сигнал/шум на бит, выраженное в децибелах.

15 Списочное декодирование полярных кодов

Необходимо реализовать процедуру построения полярных кодов для случая двоичного стирающего канала, их несистематическое кодирование, моделирование передачи и декодирование с помощью min-sum версии алгоритма Тала-Варди. Кодирование должно осуществляться как $c = u \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}^{\otimes m}$, где $u_i = 0, i \in \mathcal{F}$, $\mathcal{F}$ — множество номеров замороженных символов. Прочие элементы вектора u представляют собой последовательно выбираемые элементы кодируемого информационного вектора. Моделирование следует производить для случая канала с двоичной амплитудно-импульсной модуляцией и аддитивным белым гауссовским шумом. Под уровнем шума следует понимать отношение сигнал/шум на бит, выраженное в децибелах. Из списка, формируемого декодером Тала-Варди, необходимо выбирать наиболее вероятное кодовое слово. Входной файл на первой строке содержит длину кода 2^m, размерность k, целевую вероятность стирания в двоичном стирающем канале и размер списка в декодере Тала-Варди. Далее идут строки с командами. Первая строка выходного файла должна содержать список номеров замороженных символов (нумерация с 0), выписанных в порядке возрастания. Пример файла input.txt:

```
8 4 0.5 16
Encode 1 0 0 0
Decode -0.5 1 1 1 -1 -1 -1 -1
Simulate 3 100000 100
Simulate 4 100000 100
```

Пример файла output.txt:

```
0 1 2 4
1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 1
2.56E-2
9.31E-3
```

16 Последовательное декодирование полярных кодов

Требования аналогичны приведенным в разделе 15 с точностью до замены списочного алгоритма последовательным. При реализации следует полагать максимальное число путей (параметр D в [9]) равным $D = kL$, где k — размерность кода, L — максимальное число проходов через одну фазу. При этом следует избегать единовременного выделения памяти под все D путей.

Список литературы

[1] Мак-Вильямс, . . Теория кодов, исправляющих ошибки / Ф. Дж. Мак-Вильямс, Н. Дж. А. Слоэн. — М.: Связь, 1979.

[2] Кудряшов, . Основы теории кодирования / Б.Д. Кудряшов. — СПб: БХВ, 2016.

[3] Arıkan, E. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels / E. Arıkan // IEEE Transactions on Information Theory. — 2009. — July. — Vol. 55, no. 7. — P. 3051–3073.

[4] Guruswami, V. Improved decoding of Reed-Solomon and algebraic-geometric codes / V. Guruswami, M. Sudan // IEEE Transactions on Information Theory. — 1999. — September. — Vol. 45, no. 6. — P. 1757–1767.

[5] Nielsen, R. R. Decoding Reed-Solomon codes beyond half the minimum distance / R. R. Nielsen, T. Hoholdt // Coding theory, Cryptography and Related Areas. — Springer-Verlag, 2000. — P. 221–236.

[6] Roth, R. Efficient decoding of Reed-Solomon codes beyond half the minimum distance / R. Roth, G. Ruckenstein // IEEE Transactions on Information Theory. — 2000. — Vol. 46, no. 1. — P. 246–257.

[7] Tal, I. List decoding of polar codes / Ido Tal, A. Vardy // IEEE Transactions On Information Theory. — 2015. — May. — Vol. 61, no. 5. — P. 2213–2226.

[8] A trellis-based recursive maximum-likelihood decoding algorithm for binary linear block codes / Toru Fujiwara, Hirosuke Yamamoto, Tadao Kasami, Shu Lin // IEEE Transactions On Information Theory. — 1998. — March. — Vol. 44, no. 2.

[9] Trifonov, P. A score function for sequential decoding of polar codes / P. Trifonov // Proceedings of IEEE International Symposium on Information Theory. — Vail, USA, 2018.

[10] Valembois, A. Box and match techniques applied to soft-decision decoding / Antoine Valembois, Marc Fossorier // IEEE Transactions on Information Theory. — 2004. — 5. — Vol. 50, no. 5. — P. 796–810.

[11] Wu, Y. New list decoding algorithms for Reed-Solomon and BCH codes / Y. Wu // IEEE Transactions on Information Theory. — 2008. — August. — Vol. 54, no. 8.