

Двоичные алгоритмы сжатия на основе деревьев решений

А.Б. Левина¹, П.Г. Черненко¹, С.В. Бойко¹

¹Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»,
Россия, 197022, г. Санкт-Петербург

Аннотация. Сжатие данных – процесс преобразования данных с целью уменьшения их объёма. Двумя важными параметрами любого алгоритма сжатия являются степень сжатия и скорость. Степень сжатия имеет решающее значение, когда речь идёт о хранении или передаче больших объёмов исходных данных, в то время как скорость становится существенной при постоянном обмене небольшими объёмами данных (например, при передаче сообщений). Целью данной работы является повышение скорости сжатия данных для оптимизации использования памяти в электронных устройствах. Работа включает в себя анализ существующих методов архивирования и возможностей использования бинарных деревьев для обеспечения соответствия кода условию Фано. Разработан метод кодирования, основанный на уменьшении глубины листьев в дереве кодовых слов, а также метод, оптимизирующий глубину листьев во время кодирования путём решения задачи о рюкзаке 0-1. Для всех алгоритмов были получены теоретические оценки сложности, проведено сравнение с классическим алгоритмом Хаффмана, представлены практические реализации. Тесты показали, что разработанные алгоритмы превосходят по времени и позволяют эффективно использовать их при небольших объёмах данных, несмотря на незначительное увеличение средней длины кодового слова.

Ключевые слова: сжатие данных, степень сжатия данных, префиксные коды, оптимальный код, энтропийное кодирование, бинарные деревья, задача о рюкзаке.

Abstract. Data compression is a process of transforming data in order to reduce its volume. Two important parameters of any compression algorithm are the compression ratio and speed. The compression ratio is crucial when it comes to storing or transmitting large amounts of original data, while speed becomes significant when small amounts of data are constantly exchanged (such as in message transmission). The goal of this project is to increase the speed of data compression in order to optimize memory usage in electronic devices. The work includes an analysis of existing archiving methods and the possibilities of using binary trees to ensure that the code meets the Fano condition. A coding method based on reducing the depth of leaves in the codeword tree was developed, as well as another method that optimizes the leaf depth during coding by solving the 0-1 Knapsack Problem. Theoretical complexity estimates were obtained for all algorithms and comparisons were made with the classical Huffman algorithm. Practical implementations were also provided. The tests showed that the developed algorithms are superior in terms of time and allow for effective use on small data volumes, despite the average length of the codeword increasing slightly.

Keywords: data compression, data compression ratio, prefix codes, optimal code, entropy coding, binary tree, knapsack problem.

Введение

Теория сжатия информации, также известная как сжатие данных, берет своё начало в середине 20-го века благодаря новаторской работе Клода Шеннона. В статье 1948 года [1] Шеннон ввёл понятие энтропии, заложив основу для понимания пределов сжатия данных без потери информации. В 1970-х годах Абрахам Лемпель и Джейкоб Зив внесли значительный вклад в разработку алгоритмов Лемпеля-Зива (LZ), которые стали основой для многих современных методов сжатия, таких как ZIP и GIF [2, 3]. За прошедшие годы достижения в области технологий сжатия данных как без потерь, так и с помехами произвели революцию в области цифровых коммуникаций и мультимедиа.

Теория сжатия информации является неотъемлемой частью многих приложений, включая хранение данных, где она помогает управлять огромными объёмами информации, и телекоммуникации, где она оптимизирует передачу данных по сетям. В мультимедиа сжатие обеспечивает эффективное хранение и потоковую передачу аудио- и видеофайлов, что делает возможными такие технологии, как MP3 и MPEG [4, 5, 6]. В области машинного обучения и искусственного интеллекта методы сжатия используются для обработки больших массивов данных и повышения эффективности вычислений. Современное состояние сжатия информации продолжает развиваться благодаря совершенствованию алгоритмов и аппаратного обеспечения, что обеспечивает его важнейшую роль в управлении постоянно растущими объёмами цифровой информации. Кроме того, квантовая теория информации становится передовым направлением в понимании и использовании квантовых систем для коммуникации и вычислений [7, 8, 9].

В разделе 1 представлены математические основы, в разделе 2 проиллюстрированы созданные алгоритмы, в разделе 3 показано сравнение предложенных алгоритмов с ранее известными, а заключение приведено в разделе 4.

1. Основные теоретические принципы

В данном разделе представлены основные математические принципы, которые использованы в работе. Пусть дан алфавит C . Назовём непустой упорядоченный набор символов из алфавита кодовым словом $\bar{c}_k$. Для каждого кодового слова $\bar{c}_k$ существует определённая длина l_k , равная количеству символов в рассматриваемом кодовом слове.

Определение 1. *Префиксный код – вид кодов, обладающих следующим свойством: ни одно кодовое слово не является началом другого слова (условие Фано).*

Пусть $m = \min_{k \in \{i, j\}} \{l_k\}$, l_k – длина кодового слова $\bar{c}_k$:

$$\left. \bar{c}_k = (c_{k,s}, s \in \overline{(1, l_k)}), k \in \{i, j\} \right\} \Rightarrow \exists s \in \overline{1, m}, c_{i,s} \neq c_{j,s}. \quad (1)$$

Существование префиксных кодов определяется неравенством Крафта-Макмиллана. Каждый символ $s_1, s_2, \dots, s_n$ во входном алфавите S связан с кодовым словом $c_1, c_2, \dots, c_n$ (пусть $q = |C|$). Чтобы гарантировать, что кодовые слова не содержат префиксов, для длин кодовых слов должно выполняться следующее неравенство:

$$\sum_{i=1}^n q^{-l_i} \leq 1. \quad (2)$$

Стоит отметить, что, поскольку в работе рассматриваются только двоичные коды, то $q = 2$. Неравенство Крафта-Макмиллана является необходимым и достаточным

условием существования разделяемых и префиксных кодов с заданным набором длин кодовых слов в теории кодирования. Данное неравенство было впервые введено Леоном Крафтом в 1949 году и доказано Броксуэем Макмилланом в 1956 году. Не существует кодов, для которых длина кодового слова не соответствовала бы неравенству.

Кодовое дерево для двоичного префиксного кода – это двоичное дерево, в котором каждая вершина соответствует символу в коде, а каждый лист представляет кодовое слово. Каждый путь от корня к листу проходит по вершинам, соответствующим символам в кодовом слове в том порядке, в котором они появляются в слове.

Определение 2. *Задана дискретная случайная величина (ДСВ) ξ , принимающая значения из набора X и распределяется в соответствии с $p: X \rightarrow [0, 1]$, тогда энтропия:*

$$H(\xi) = - \sum_{x \in X} p(x) \log_2 p(x). \quad (3)$$

Определение 3. *Задана дискретная случайная величина (ДСВ) ξ , принимающая значения из набора X и распределяется в соответствии с $p: X \rightarrow [0, 1]$, $l: X \rightarrow \mathbb{N}$ функция длин кодовых слов, тогда средняя длина:*

$$L(\xi) = \sum_{x \in X} p(x) l(x). \quad (4)$$

Определение 4. *Средняя длина является оптимальной, если она минимальна для данного распределения входных данных $\bar{p} = (p_1, p_2, \dots, p_n)$ (где p_i – вероятность символа s_i), обозначим как L^* .*

$$L^*(\xi) = \sum_{x \in X} p(x) l(x) \rightarrow \min. \quad (5)$$

Поскольку средняя ожидаемая длина кодового слова L задаёт среднюю длину, именно это значение пытаются уменьшить. Нижняя граница данного значения – энтропия, на чём основано кодирование Шеннона.

Теорема 1. *Пусть задана ДСВ, моделирующее источник сообщения $\bar{p} = (p_i, i \in \overline{1, n})$ и оптимальный префиксный код с назначенной длиной кодового слова $\bar{l} = (l_i, i \in \overline{1, n})$ используется для кодирования. Тогда:*

$$\forall \xi \quad H(\xi) \leq L^*(\xi) \leq H(\xi) + 1. \quad (6)$$

Доказательство. Мы отдельно докажем нижнюю и верхнюю границы оценки.

1) Нижняя граница

Пусть $Q = \sum_{i=1}^n 2^{-l_i}$; Из неравенства Крафта $Q \leq 1$, тогда $\log_2 Q < 0$.

Пусть $q_i = \frac{2^{-l_i}}{Q} \in (0; 1)$; $\sum_{i=1}^n q_i = \frac{1}{Q} \sum_{i=1}^n 2^{-l_i} = \frac{Q}{Q} = 1$.

$$\begin{aligned} H(\xi) &= - \sum_{i=1}^n p_i \log_2 p_i \leq - \sum_{i=1}^n p_i \log_2 q = - \sum_{i=1}^n p_i \log_2 \frac{2^{-l_i}}{Q} = \\ &= \sum_{i=1}^n p_i \log_2 Q - \sum_{i=1}^n p_i \log_2 2^{-l_i} = \log_2 Q \sum_{i=1}^n p_i + \sum_{i=1}^n p_i l_i \leq L^*. \end{aligned}$$

2) Верхняя граница

Пусть $l'_i = \lceil -\log_2 p_i \rceil$; $-\log_2 p_i < l'_i \leq -\log_2 p_i + 1$.

$$\frac{1}{p_i} = 2^{-\log_2 p_i} < 2^{l'_i}; p_i > 2^{-l_i}; 1 = \sum_{i=1}^n p_i > \sum_{i=1}^n 2^{-l'_i}.$$

$\exists c'$ – префиксный код с длиной кодового слова l'_i ;

$$L^* = \sum_{i=1}^n p_i l_i \leq \sum_{i=1}^n p_i l'_i \leq \sum_{i=1}^n p_i (-\log_2 p_i + 1) = - \sum_{i=1}^n p_i \log_2 p_i + \sum_{i=1}^n p_i =$$

$= H(\xi) + 1$, тогда $\forall \xi \ H(\xi) \leq L^*(\xi) \leq H(\xi) + 1$, что и требовалось доказать.

1.1 Алгоритм Шеннона

Все алгоритмы, которые будут рассмотрены в данной статье, получают множество пар (символ s , вероятность символа p_s), отсортированных в порядке убывания, после чего возвращают множество пар (символ s , длина кодового слова l_s), которое также сортируется в порядке убывания:

$$\{(s_1, p_1), (s_2, p_2), \dots, (s_n, p_n)\} \ \forall i \leq j, \ i, j \in \overline{1, n}, \ p_i \geq p_j. \quad (7)$$

В системе кодирования Шеннона [10] символы упорядочены от наиболее вероятных к наименее вероятным, а кодовые слова присваиваются путём взятия первой цифры из двоичного разложения совокупных вероятностей:

$$l_i = \lceil -\log_2 p_i \rceil. \quad (8)$$

Таким образом, остаётся выбрать кодограмму заранее определённой длины для каждого из символов алфавита, соблюдая условие префикса. После выполнения алгоритма генерируется таблица с символами и соответствующими им кодами. На основе полученной таблицы можно построить дерево двоичных кодов (рис. 1).

symbol	codeword length	codeword
a	1	0
b	2	10
c	3	110

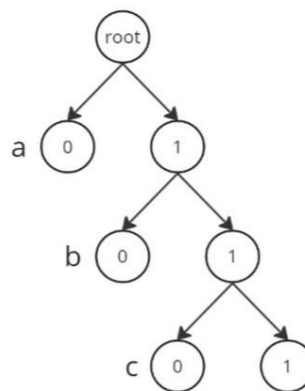


Рисунок 1 – Пример кодовой таблицы и двоичного кодового дерева

Изменяя структуру дерева, можно уменьшить глубину расположения узлов (рис. 2).

symbol	codeword length	codeword
a	1	0
b	2	10
c	2	11

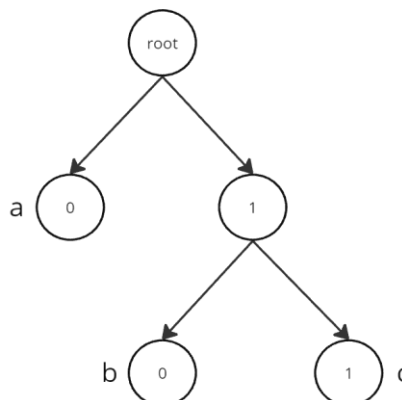


Рисунок 2 – Пример кодовой таблицы после перестройки

1.2 Алгоритмы класса LZ

Семейство алгоритмов сжатия Lempel-Ziv (LZ) произвело революцию в методах сжатия данных, внедрив новый подход к кодированию повторяющихся последовательностей. В основе алгоритмов лежит идея использования скользящего окна, которое поддерживает динамическое кодовое дерево ранее встреченных последовательностей. При проверке новой последовательности алгоритм ищет наиболее длинное совпадение в существующих данных, хранящихся в окне. Вместо сохранения повторяющихся последовательностей он заменяет их указателями, соответствующими положению и длине совпадающего сегмента, что не только сокращает объем данных, но и устраняет избыточность, присущую большинству типов данных.

Одной из ключевых концепций семейства LZ является использование кодов переменной длины для представления последовательностей. По мере того, как алгоритм просматривает входные данные, создаётся словарь уникальных последовательностей, увеличивающийся по мере обработки большего количества данных. Для каждого нового фрагмента данных алгоритм проверяет словарь на наличие существующих последовательностей и кодирует данные либо как ссылку на словарную статью, либо как новую запись – подобная стратегия кодирования однотипных фрагментов данных позволяет значительно уменьшить размер файла, особенно при работе с повторяющимися шаблонами, такими как текстовые файлы или определённые типы изображений.

Эффективность алгоритмов LZ обусловлена их способностью адаптивно обновлять дерево кодов за счёт сохранения наиболее часто используемых последовательностей в более коротких кодах. По мере обработки данных выявляются и сжимаются распространённые шаблоны, в то время как менее распространённые могут оставаться несжатыми. Подобная динамическая настройка не только оптимизирует степень сжатия, но и ускоряет процесс декодирования, так как используемые последовательности могут быть быстро извлечены из дерева кода.

Семейство алгоритмов LZ является примером того, как распознавание и использование избыточности в данных может привести к созданию мощных и эффективных методов сжатия. Сосредоточив внимание на шаблонах и используя механизм скользящего окна для создания постоянно расширяющегося словаря, алгоритмы LZ достигают впечатляющих показателей сжатия, что делает их незаменимыми в различных приложениях, от архивации файлов до передачи данных по сетям [11].

1.3 Случайные ориентированные ациклические графы решений: Энтропийное сжатие для случайных лесов (RDAG)

Рассмотрим несколько современных алгоритмов сжатия, основанных на сжатии деревьев кода. Данный алгоритм использует древовидные ансамбли, такие как Random Forest (RF) – популярный метод машинного обучения из-за его вычислительной эффективности. Однако при работе с большими данными древовидные ансамбли часто разрастаются до больших размеров, что вызывает ограничения, связанные с устройствами с небольшим объёмом памяти (например, мобильными телефонами). В этом случае соответствующие деревья обрезаются до состояния, при котором можно восстановить вес леса путём обрезки.

Особенность RDAG заключается в том, что риск переобучения возрастает с каждым разделением узлов в дереве. Предварительная обрезка может помочь снизить этот риск, предотвращая ненужное разделение узлов, в то время как последующая обрезка (или

2.1 Алгоритм сжатия с оптимизацией глубины листа (CAOL)

Рассмотрим в качестве примера двоичное дерево, которое алгоритм получает в качестве входных данных (рис. 3).

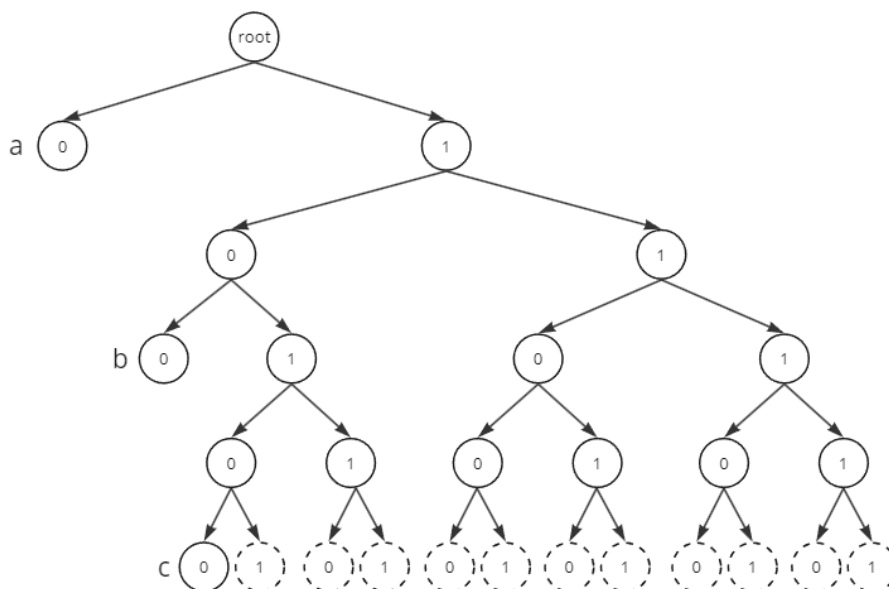


Рисунок 3 – Двоичное дерево на входе алгоритма

Пунктирные линии на графе указывают на листья, которые не используются никакими символами. Для удобства заменим понятие длины кодового слова l_k на глубину листа d_k , так как они эквивалентны.

Поскольку длина кодового слова зависит от вероятности, оптимизируем листы с наименьшей глубиной. Пусть изменена глубина символов, тогда длина их новых кодовых слов равна l'_s – это уменьшает длину на $k_s = l_s - l'_s$, а ожидаемая длина кодового слова изменится на $p_s k_s$. Важно заметить, что уменьшить длину кодового слова с 1 до 0 невозможно, поэтому листы на глубине 1 будут пропущены (символ «а»).

Уменьшим глубину следующего символа «b», который имеет глубину 3. Чтобы уменьшить глубину, нужен ещё как минимум 1 свободный узел с глубиной 3 – такие узлы в графе есть, поэтому его можно перестроить.

Пусть d_{max} – максимальная глубина листьев. Чтобы сохранить условие префикса, узел глубины $(d_{max} - 1)$ использует 2 узла глубины d_{max} . Узел глубины $(d_{max} - 2)$ использует 4 узла глубины d_{max} . Если выберем узел с произвольной глубиной (длина кодового слова равна глубине листа), равной d_s , то для того, чтобы уменьшить его глубину на k_s , должно быть доступно W узлов с глубиной d_{max} , где W :

$$W(d_{max}, d_s, k_s) = 2^{d_{max}-d_s} (2^{k_s} - 1). \quad (9)$$

Алгоритм. В начале алгоритма подсчитываются все используемые узлы. Обозначим их через U .

Для всех $s \in X: k_s = 1; U = 2^{d_{max}} - \sum 2^{d_{max}-d_s},$

При таких вероятностях остаётся $U = 2$. Если оптимизируется символ «А», то средняя длина кодового слова уменьшится на $k_A l_A = 0,3$ бита. Если оптимизируется как «В», так и «С», то средняя длина уменьшится на $k_B * l_B + k_C l_C = 0,4$ бита.

Чтобы решить описанную проблему с помощью компьютерных алгоритмов, можно её свести к варианту задачи динамического программирования «рюкзак». Используя методы динамического программирования, задачу можно решить за $O(C \times T)$, где C – вместимость, T – общее количество элементов.

В подобном сценарии U является рюкзаком, $W(d_{max}, d_s, k_s)$ – вес предметов, а p_s – стоимость. Для одного объекта может быть выполнено несколько оптимизаций, но одновременно может быть применена только одна. Например, невозможно уменьшить длину кодового слова одновременно на 1 и 2 бита. Сложность рассматриваемого алгоритма полностью зависит от сложности решения задачи о рюкзаке 0-1 (рисунок 6).

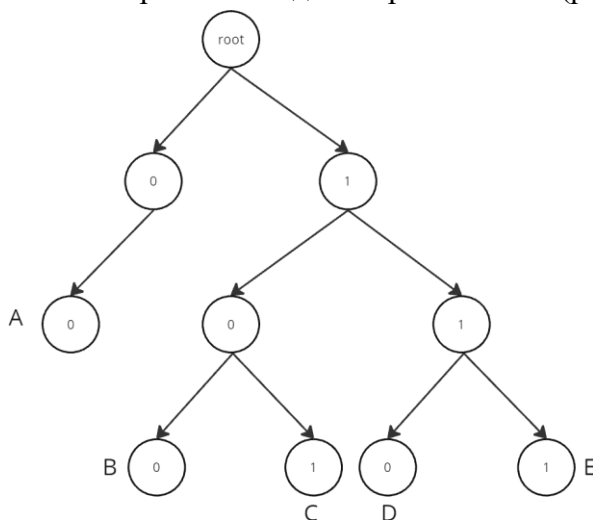


Рисунок 6 – Пример кодового дерева для кода из Таблицы 1

3. Сравнение предложенных алгоритмов с алгоритмом кодирования Хаффмана и алгоритмом Шеннона

3.1 Вычислительная сложность

Для выполнения алгоритма сжатия с оптимизацией глубины листа требуется $O(N)$ времени и памяти – это связано с тем, что оба алгоритма выполняют ряд математических операций над каждым символом алфавита.

Алгоритм на основе рюкзака использует выходные данные алгоритма Шеннона и создаёт таблицу. Пусть k_{max} – количество возможностей оптимизации для слова с наименьшей вероятностью. Размер таблицы ограничен произведением $N \times U \times k_{max}$, так как для каждого символа может быть не более k_{max} оптимизаций, и для каждого необходимо выполнить итерацию по всем неиспользуемым узлам от 0 до U (таблица 2).

Таблица 2 – Сравнение вычислительной сложности

Алгоритм	Временная сложность	Пространственная сложность
Алгоритм кодирования Хаффмана	$O(N)$	$O(N)$
Алгоритм Шеннона	$O(N)$	$O(N)$
CAOL	$O(N)$	$O(N)$
Алгоритм на задаче о рюкзаке	$O(N \times U \times k_{max})$	$O(N \times U \times k_{max})$

3.2 Сравнение программной реализации

Все алгоритмы были протестированы на текстах размером 138 байт, 9,2 КБ и 4,2 МБ для имитации сжатия небольших, средних и больших текстов. Из-за отсутствия достаточных ресурсов не удалось представить результаты сжатия большого файла с использованием третьего алгоритма. Исходя из плана исследований, реализация алгоритма будет оптимизирована. Результаты приведены в таблице 3, где γ – средняя длина закодированного сообщения по сравнению с исходным сообщением в процентах.

Таблица 3 – Сравнение программной реализации

Алгоритм	Время работы, μ s	γ
Алгоритм кодирования Хаффмана	143,6 / 252,3 / 344	53,62 / 53,76 / 57,91
Алгоритм Шеннона	78,2 / 142,2 / 204	58,88 / 57,34 / 62,94
CAOL	69,2 / 110,2 / 175	55,16 / 54,33 / 58,89

Было обнаружено, что алгоритм сжатия с оптимизированной глубиной листа в два раза быстрее классического алгоритма Хаффмана. Однако это привело к снижению степени сжатия на 5 % по сравнению с классическим алгоритмом и на 1 % по сравнению с другим оптимизированным алгоритмом. Полученные результаты позволяют предположить, что данные алгоритмы могут быть более подходящими для сжатия большого количества небольших сообщений, где скорость передачи важнее степени сжатия.

Заключение

В результате исследования были разработаны два алгоритма сжатия данных и проведено теоретическое сравнение с классическими алгоритмами. Представлена реализация одного из этих алгоритмов, а также практическое сравнение. Установлено, что алгоритм CAOL в два раза быстрее классического алгоритма Хаффмана, но на 1 % менее эффективен с точки зрения степени сжатия – это означает, что CAOL обрабатывает данные значительно быстрее, что может иметь решающее значение в контексте скорости передачи данных. Небольшой компромисс в эффективности сжатия может оказаться приемлемым при работе с большим количеством небольших сообщений – в подобных случаях скорость передачи данных важнее степени сжатия, что делает алгоритм CAOL предпочтительным выбором. Рассмотренные алгоритмы могут быть полезны в системах, которым необходимо быстро обрабатывать и передавать многочисленные короткие сообщения.

С точки зрения приложений, алгоритм CAOL применим в системах связи в режиме реального времени, таких как приложения для обмена сообщениями или онлайн-игры, в которых важна быстрая передача данных. Кроме того, алгоритм также может быть полезен в сетях Интернета вещей (IoT), где между устройствами постоянно передаётся множество небольших пакетов данных.

Возможные усовершенствования алгоритма CAOL могли бы включать методы оптимизации для повышения эффективности сжатия без ущерба для скорости. Кроме того, можно было бы изучить гибридные подходы, сочетающие CAOL с другими алгоритмами для более эффективного баланса скорости и степени сжатия. Другим способом улучшения может быть адаптация алгоритма к конкретным типам данных или коммуникационным протоколам для дальнейшего повышения производительности в целевых приложениях.

Эффективность алгоритма, основанного на задаче о рюкзаке 0-1, определяется используемым методом решения, а не особенностями реализации. Первоначальная задача поиска оптимального решения переформулирована в терминах задачи о рюкзаке, поэтому эффективность зависит от точности решения.

В настоящее время методы динамического программирования являются самым быстрым способом решения задачи, однако также существуют эвристические методы, обеспечивающие хорошую аппроксимацию оптимального решения, что может быть полезно для ускорения работы. Данные методы не гарантируют оптимального решения, но могут обеспечить разумное приближение за более короткое время.

Благодарности

Работа выполнена в рамках государственного задания Министерства науки и высшего образования Российской Федерации № 075-00003-24-02 (FSEE-2024-0003).

Список литературы

1. Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3), 379-423;
2. Bell, T. C., Cleary, J. G., & Witten, I. H. (1990). Text Compression. Prentice-Hall, Englewood Cliffs;
3. Storer, J. A. (1987). Data Compression: methods and theory. Computer Science Press, Rockville;
4. Brandenburg, K. (1997). MP3 and AAC Explained. In: *The Digital Signal Processing Handbook*. CRC Press, Boca Raton;
5. Le Gall, D. (1991). MPEG: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4), 46-58;
6. Rao, K. R., Bojkovic, Z. S., & Milovanovic, D. A. (2002). Multimedia communication systems: techniques, standards, and networks. Prentice Hall PTR;
7. Nielsen, M. A., & Chuang, I. L. (2000). Quantum Computation and Quantum Information. Cambridge University Press, Cambridge;
8. Schumacher, B., & Westmoreland, M. D. (2010). Quantum Processes, Systems, and Information. Cambridge University Press, Cambridge;
9. Wilde, M. M. (2017). Quantum Information Theory. 2nd edn. Cambridge University Press, Cambridge;
10. Yaglom, A. M., & Yaglom, I. M. (1983). Probability and Information. Nauka, Moscow;
11. Ziv, J., & Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3), 337-343;
12. Liu, X., Liu, X., Lai, Y., Yang, F., & Zeng, Y. (2019). Random decision DAG: an entropy based compression approach for random forest. In *International Conference on Database Systems for Advanced Applications*. DASFAA 2019, LNCS 11448, pp. 319–323. DOI: 10.1007/978-3-030-18590-9_37.