Об одном алгоритме сжатия данных

К.В. Харин

1 Введение

Проблема сжатия данных является одной из важнейших задач теории кодирования и ее приложений. Основным теоретическим результатом в этой области является теорема Хаффмана о кодах с минимальной избыточностью (см., например, [?]). Использование этой теоремы в большинстве алгоритмов сжатия данных заключается в построении словаря повторяющихся фрагментов входной последовательности данных и последующем кодировании таких фрагментов кодовыми словами в выходном алфавите так, чтобы наиболее длинным или часто встречающимся фрагментам соответствовали более короткие кодовые слова. При этом строящийся код должен будет обладать свойством префиксности для обеспечения однозначности декодирования.

Реализация этого подхода связана с решением сложных переборных задач и с большими затратами памяти для хранения словаря и кодовых слов. В настоящей работе предлагается алгоритм, основанный на другом подходе к решению задачи сжатия данных. Входная последовательность разбивается на слова определенной небольшой длины, после чего наборы таких слов записываются в виде формул в специально подобранном базисе. Информация о базисе также записывается в выходной последовательности и декодирование осуществляется путем подстановки элементов этого базиса в формулу для восстановления исходных данных.

На наш взгляд предлагаемый алгоритм обладает рядом преимуществ перед традиционными, поскольку при кодировании нет необходимости в предварительном просмотре входной последовательности и решении переборных задач построения словаря, а декодирование осуществляется максимально просто без существенных затрат памяти.

Прежде чем собственно строить алгоритм сформулируем строго решаемую задачу.

В терминах машин Тьюринга задача сжатия данных может быть сформулирована следующим образом. Имеется входная лента, которую можно считать ограниченной с левой стороны. На ленте могут записываться символы из алфавита $\{0, 1, *, \#\}$. В начальный момент времени t_0 на ленте, начиная с левого конца, записана последовательность символов $(*, a_0, a_1, \ldots, a_{L-1}, a_L)$, где $a_0, \ldots, a_L \in \{0, 1\}$ - элементы входной двоичной последовательности, подлежащей сжатию. Головка машины в начальном состоянии q_0 устанавливается в крайнюю левую ячейку (с символом $'*'$).

Требуется построить такую машину Тьюринга (алгоритм) $\mathfrak{A}$, чтобы она, начав работу в описанной конфигурации, через конечное число шагов остановилась, перейдя в заключительное состояние и оставив на ленте одну из следующих последовательностей символов:

а) $(\#, b_0, b_1, \ldots, b_{L_1})$, где $b_0, \ldots, b_{L_1} \in \{0, 1\}$, и $L_1 < L$,

либо

б) $(*, a_0, a_1, \ldots, a_L)$, то есть повторить входную последовательность, что в данном случае будет означать отказ от сжатия.

Кроме того, требуемая машина Тьюринга при подаче на ее вход в начальный момент времени в начальном состоянии последовательности $(\#, b_0, b_1, \ldots, b_{L_1})$, записанной на ленте, начиная с левого конца, через конечное число шагов должна остановиться, перейдя в заключительное состояние, после чего на ленте должна быть записана исходная последовательность $(*, a_0, a_1, \ldots, a_L)$, что и будет, собственно, означать возможность точного восстановления исходных данных по сжатой последовательности (условие "сжатия без потерь").

Вспомогательные символы алфавита '$*$' и '$\#$' служат только для задания "направления"работы алгоритма (сжатие или восстановление) и введены исключительно для удобства. Вместо них вполне можно использовать символы '0' и '1'. Условие на длину выходной последовательности $L_1 < L$ означает именно сжатие (уменьшение длины) входной последовательности при сохранении возможности ее восстановления.

Для решения этой задачи в настоящей работе предлагается алгоритм, основная идея которого заключается в разбиении входной последовательности на слова определенной небольшой длины и записи

наборов таких слов в виде формул в специально подобранном базисе.

2 Основные понятия и определения

В этом разделе будут введены некоторые понятия, которые будут использоваться при описании и обосновании алгоритма.

Пусть $n \in \mathbb{N},\ n \geq 2$. На множестве $E_n = \{0, \ldots, n-1\}$ рассмотрим произвольное отношение эквивалентности σ, то есть разобьем E_n на N_σ непересекающихся классов эквивалентности. Эквивалентность элементов i и j множества E_n будем обозначать $i \overset{\sigma}{\sim} j$. Рассмотрим теперь произвольное слово A длины n в алфавите $\{0,1\}$, то есть $A \in E_2^n$. $A = (a_0, \ldots, a_{n-1})$. Множество E_n можно рассматривать как множество индексов букв слова A.

Определение 1. Отношение эквивалентности σ на множестве E_n называется согласованным со словом $A = (a_0, \ldots, a_{n-1}) \in E_2^n$, если для любых $i_1, i_2 \in E_n$ из $i_1 \overset{\sigma}{\sim} i_2$ следует, что $a_{i_1} = a_{i_2}$.

Таким образом, буквы слова A, индексы которых эквивалентны, должны совпадать.

Пример. Пусть в множестве $E_4 = \{1,2,3,4\}$ задано отношение эквивалентности σ, такое что $1 \overset{\sigma}{\sim} 2$, $3 \overset{\sigma}{\sim} 4$, но $2 \overset{\sigma}{\not\sim} 3$, то есть классами эквивалентности относительно σ являются подмножества $\{1,2\}$ и $\{3,4\}$. Тогда отношение σ будет согласовано со следующими словами из E_2^4: (0000), (0011), (1100), (1111) и не будет согласовано с остальными 12 элементами E_2^4.

Пусть теперь $\mathcal{A} = \langle A^1, \ldots, A^k \rangle$ - набор слов из E_2^n.

Определение 2. Отношение эквивалентности σ на множестве E_n оптимально согласовано с набором $\mathcal{A}$, если:

1. σ согласовано с A^i для любого $i \in \{1, \ldots, k\}$.

2. Для любого отношения эквивалентности τ на E_n, обладающего свойством 1, количество N_τ классов эквивалентности, на которые разбивается E_n в соответствии с τ, не меньше количества N_σ классов эквивалентности для отношения σ ($N_\tau \geq N_\sigma$).

Необходимо заметить, что отношение эквивалентности, разбивающее E_n на n классов (то есть отношение, при котором никакие два элемента $i, j \in E_n$ при $i \neq j$ не эквивалентны), согласовано с любым словом A из E_2^n. Таким образом, свойство 2 из определения 2 действительно

задает в некотором смысле оптимальность выбранного отношения эквивалентности. Кроме того, любое отношение эквивалентности τ, согласованное со всеми элементами набора $\mathcal{A}$, задает разбиение на классы эквивалентности, которое является подразбиением соответствующего разбиения для оптимально согласованного отношения эквивалентности. Учитывая сказанное, а также определения 1 и 2, можно показать истинность следующего утверждения.

Утверждение 1 *Для любого набора слов* $\mathcal{A} = \langle A^1, \ldots, A^k \rangle$ *из* E_2^n *существует оптимально согласованное с* $\mathcal{A}$ *отношение эквивалентности на множестве индексов* E_n*, причем такое отношение единственно.*

3 Основная идея алгоритма

Пусть имеется входная последовательность данных $(a_0, a_1, ..., a_L)$, где $a_i \in \{0, 1\}$ при $i = 0, \ldots, L$. Выберем натуральное число n в качестве параметра алгоритма (из практических соображений n удобно выбирать из значений вида $n = 2^q$, где $q = 3, 4, 5, 6$). Рассмотрим последовательно идущие слова входной последовательности длины n $A^1 = (a_0, \ldots, a_{n-1}), A^2 = (a_n, \ldots, a_{2n-1}), \ldots$ Можно считать, что $L + 1 = b \cdot n$ $(b \in \mathbb{N})$, при необходимости дополняя исходную последовательность нулями.

Будем строить отношения эквивалентности $\sigma_1, \sigma_2, \cdots$ на E_n, такие что

$$\begin{array}{ll} \sigma_1 & \text{оптимально согласовано с } \langle A^1 \rangle \\ \sigma_2 & \text{оптимально согласовано с } \langle A^1, A^2 \rangle \\ \vdots & \\ \sigma_k & \text{оптимально согласовано с } \langle A^1, A^2, \ldots, A^k \rangle \\ \vdots & \end{array}$$

Обозначим количество классов эквивалентности, на которые разбивается E_n в соответствии с отношением эквивалентности σ_i через N_{σ_i}.

Процесс построения отношений $\sigma_1, \sigma_2, \ldots$ можно описать с использованием индукции:

а) Слову A^1 в общем случае соответствует разбиение t_1 на два класса индексов из E_n: индексы, соответствующие буквам, принимающим

значение 0 в A^1, и буквам, принимающим значение 1. Это разбиение выбирается в качестве разбиения T_{σ_1}, соответствующего отношению эквивалентности σ_1, согласованному с A_1.

б) Если построено разбиение T_{σ_k} множества E_n на классы эквивалентности для отношения σ_k, оптимально согласованного с набором $\langle A^1, \ldots, A^k \rangle$, то рассмотрим слово A^{k+1}, которое задает разбиение t_{k+1} множества E_n на два класса, аналогичное рассмотренному в пункте а). В качестве разбиения $T_{\sigma_{k+1}}$, соответствующего отношению эквивалентности σ_{k+1} выбирается подразбиение разбиения T_{σ_k}, полученное его пересечением с разбиением t_{k+1}.

Очевидно, что процесс построения отношений $\sigma_1, \ldots, \sigma_k \ldots$ необходимо продолжать, пока $N_{\sigma_k} < n$, и не исчерпана вся входная последовательность. Пусть l таково, что $N_{\sigma_l} < n$, $N_{\sigma_{l+1}} = n$, либо $L = ln$.

Для любого $i \in \{1, \ldots, l\}$ можно записать всю информацию, содержащуюся в элементах набора $\langle A^1, \ldots, A^i \rangle$, следующим образом. Выходная последовательность будет состоять из трех частей:

$$[\text{ запись } \sigma_i][\text{ запись } i][\underbrace{(\quad)(\quad)\ldots(\quad)}_{\substack{i \text{ записей, соответствующих} \\ \text{элементам } A^j (1 \le j \le i)}}]$$

Занумеруем классы эквивалентности на E_n по отношению σ_i числами от 0 до $N_{\sigma_i} - 1$ так, чтобы первый элемент множества E_n принадлежал классу с номером 0. Запись σ_i (то есть слово, кодирующее всю информацию об отношении эквивалентности σ_i) может быть составлена из $n-1$ двоичных чисел – номеров классов эквивалентности, которым принадлежат элементы $1, \ldots, n-1$ множества E_n. Длина такой записи равна $(n-1) \cdot] \log_2 N_{\sigma_i} [$.

Запись i представляет собой просто двоичную запись числа i. Каждая из i записей, соответствующих A^j $(j = 1, \ldots, i)$, представляет собой двоичное слово длины N_{σ_i}, элементы которого являются значениями букв a^j_m, соответствующих произвольному представителю m каждого из N_{σ_i} классов эквивалентности на множестве индексов E_n в соответствии с выбранной выше нумерацией.

Таким образом, длина всего выходного слова будет равна $l^1_i = (n-1) \cdot] \log_2 N_{\sigma_i} [+] \log_2 i [+ i \cdot N_{\sigma_i}$.

Учитывая, что длина части входной последовательности, соответствующей набору слов $\langle A^1, \ldots, A^i \rangle$ равна $l_i = in$, можно

вычислить коэффициент сжатия $\lambda_i = l_i/l_i^1$. Если $\lambda_i > 1$, то часть входной последовательности $(a_0, \ldots, a_{in-1})$ может быть сжата.

Следует отметить, что сжатие подобным образом можно рассматривать как запись каждого из слов набора $\langle A^1, \ldots, A^i \rangle$ в виде формулы в специально построенном для этого набора базисе (точнее, в полной системе). Для понимания этой идеи следует привести некоторые понятия и определения из работы [?]. А именно, рассмотрим функции вида $f : E_n \to E_2$ и обозначим все множество таких функций через P_{E_n,E_2}. Введем на P_{E_n,E_2} операции конъюнкции, дизъюнкции и отрицания следующим образом: для любых $f, g \in P_{E_n,E_2}$ и $x \in E_n$ положим

$(f\&g)(x) = f(x)\&g(x)$,
$(f \vee g)(x) = f(x) \vee g(x)$,
$(\neg f)(x) = \neg(f(x))$,

где операции $\&, \vee, \neg$ над значениями функций f и g в точке x – элементами множества E_2 – понимаются в традиционном смысле. Таким образом, операции на P_{E_n,E_2} определяются поточечно. В алгебре $\langle P_{E_n,E_2}, \Omega \rangle$, где $\Omega = \{\&, \vee, \neg\}$ можно ввести понятия полной системы и базиса следующим образом. Множество $B \subseteq P_{E_n,E_2}$ называется полной системой, если $[B]_\Omega = P_{E_n,E_2}$. Базисом называется такая полная система, удаление из которой любого элемента делает ее неполной. Можно также по аналогии рассматривать полные системы и базисы для подмножеств P_{E_n,E_2}.

В данном случае для набора слов $\langle A^1, \ldots, A^i \rangle$ по сути строится полная система, состоящая из характеристических функций классов эквивалентности по отношению σ_i на множестве индексов E_n. Эти функции можно рассматривать и как элементы P_{E_n,E_2}, и как слова из E_2^n. Слова $A^1, \ldots, A^i$ строятся из элементов этой полной системы с помощью введенных выше операций дизъюнкции и отрицания в соответствии со значениями букв этих слов на каждом из классов эквивалентности.

Итак, выше нами была описана процедура построения фрагмента сжатой выходной последовательности, соответствующей набору входных слов $\langle A^1, \ldots, A^i \rangle$. Очевидно, что таких наборов, для которых можно построить сжатую последовательность, может быть много, причем они могут пересекаться друг с другом, то есть часть одного "сжимаемого"набора может быть частью другого. В этом случае для исключения помещения лишней информации в выходную

последовательность, которая должна быть как можно короче, для сжатия выбирается только часть "сжимаемых"наборов, так что никакие два выбранных набора не пересекаются. Вопросы оптимизации такого выбора наборов и другие вопросы, связанные с оптимизацией описанного алгоритма, будут затронуты ниже.

Остается упомянуть о полной структуре выходной сжатой последовательности с учетом наличия нескольких сжимаемых наборов входных слов. Эта структура такова:

$$[(*)][x_1][a_0,\ldots,a_{x_1-1}][\begin{smallmatrix}\text{сжатая последовательность}\\ \text{для набора } \langle A^{i_1},\ldots,A^{i_1+p_1}\rangle\end{smallmatrix}][x_2][a_{j_1},\ldots,a_{j_1+x_2-1}]$$

$$[\begin{smallmatrix}\text{сжатая последовательность}\\ \text{для набора } \langle A^{i_2},\ldots,A^{i_2+p_2}\rangle\end{smallmatrix}]\cdots \qquad \cdots[x^*][a_{j_p},\ldots,a_L],$$

где $(*)$ обозначает определенную последовательность символов, обозначающую, является ли данная последовательность сжатой (при отказе от сжатия выходная последовательность начинается с символов $(\#)$, после чего переписывается вся входная последовательность), x_q обозначает номер буквы, с которой начинается q-й сжимаемый набор слов. Фрагменты, содержащие буквы входной последовательности a_{j_s}, соответствуют несжимаемым ее частям, x^* обозначает определенную последовательность символов, которая завершает последнюю сжатую подпоследовательность, после которой следует только несжимаемый фрагмент входной последовательности.

4 Возможные пути оптимизации алгоритма

В предыдущем разделе была изложена основная идея алгоритма решения задачи сжатия данных и было показано, как могут быть выделены части входной последовательности, для которых сжатие возможно. Необходимо обратить внимание на два момента, позволяющих в некоторой степени управлять процессом отбора сжимаемых наборов слов входной последовательности.

1. Рассмотрим набор $\langle A^1,\ldots,A^l\rangle$, где l таково, что $N_{\sigma_l} < n$, $N_{\sigma_{l+1}} = n$, либо $L = ln$. В этом случае обозначим через I множество $I \subset \{1,\ldots,l\}$, такое что для любого $i \in I$ $\lambda_i > 1$, где λ_i - коэффициент сжатия, вычисленный для набора $\langle A^1,\ldots,A^i\rangle$. При этом если $|I| >$

1, то существует несколько наборов слов входной последовательности, начинающихся со слова A^1, для которых возможно сжатие.

2. Как уже было отмечено ранее, различные "сжимаемые"наборы слов входной последовательности могут пересекаться друг с другом. При этом, как и в предыдущем случае, различные наборы могут иметь разный коэффициент сжатия и разную длину, что может сильно влиять на общую степень сжатия всей последовательности.

В обоих приведенных случаях для исключения помещения лишней информации в выходную последовательность и для обеспечения лучшего сжатия данных имело бы смысл воспользоваться каким-либо критерием оптимальности "сжимаемого"набора слов, позволяющего выделить наилучший в некотором смысле набор из всех, имеющих общие элементы. Ряд таких критериев может, например, возникнуть из содержательной сути решаемой задачи сжатия.

В качестве возможных критериев можно рассмотреть следующие числовые характеристики: коэффициент сжатия λ_i, произведение $\lambda_i \cdot i$ коэффициента сжатия на длину набора и некоторые другие комбинации параметров "сжимаемого"набора.

Поскольку аналитическое и даже статистическое исследование характеристик данного алгоритма представляется достаточно трудоемким из-за многостороннего влияния различных факторов и структуры входной последовательности на степень сжатия, были проведены модельные исследования алгоритма с различными критериями оптимальности. Алгоритм был реализован в виде программы на языке C++ и был испытан на реальных входных данных (текстовых, графических, файлах баз данных и т.д.). При этом чаще всего лучшие результаты демонстрировались в случае, когда в качестве критерия оптимальности набора выбиралось произведение $\lambda_i \cdot i$.

В заключение автор выражает благодарность Д.Н. Бабину и А.С. Строгалову за поддержку работы и плодотворное участие в обсуждениях.

Список литературы

[1] Яблонский С.В. Введение в дискретную математику. – М., Наука. 1986.

[2] Харин К.В. Об одной алгебре изображений // Дискретная математика. – 1996. – Т. 8, вып. 4. – С. 149-156.