УДК 621.391. ...

©2001 г. **Д.А. Шкарин**

ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ АЛГОРИТМА PPM

Для алгоритма PPM предложены новые определения обобщенных частот символов и "уходов", позволившие заметно увеличить эффективность кодирования (сжатия) различных типов реальных дискретных данных без искажений.

§ 1. Введение

PPM – один из наиболее многообещающих алгоритмов сжатия дискретных данных без искажений, использующий марковскую модель порядка D. Его основная особенность — кодирование нового (в заданном контексте) символа в одном из внутренних узлов контекстного дерева; для описания этого узла используются специальные символы ухода. На практике большая часть символов кодируется во внутренних узлах дерева и марковская модель становится достаточно условной.

Для повышения эффективности алгоритма PPM в настоящей работе предложена его модификация — PPMII (PPM with Information Inheritance). В § 2 кратко излагается алгоритм PPM и вводятся необходимые определения. В § 3 введено новое определение обобщенных частот символов (с учетом "унаследованных частот"), а в § 4 — новое определение адаптивной оценки обобщенных частот ухода и вероятностей ухода; в § 5 описывается несколько приемов уменьшения вычислительной сложности алгоритма за счет некоторого снижения эффективности сжатия. В § 6 приведены экспериментальные результаты и проведено сравнение основных характеристик PPMII как с наиболее эффективными из схем сжатия, описанных в литературе, так и с наиболее распространенными практическими реализациями алгоритмов сжатия.

§ 2. Основные обозначения. Алгоритм PPM

Пусть A — дискретный алфавит из $M \geq 2$ символов; $x^n = x_1, \ldots, x_n$, где $x_i \in A$ — первые n символов кодируемого сообщения; ω — вероятностный источник сообщений; $|\sigma|$ — длина последовательности σ или мощность множества σ. Арифметическое последовательное кодирование (сжатие) определяется *кодовыми* распределениями вероятностей $\{Q(x^n),\ x^n \in A^n\}$, $n = 1, 2, \ldots$, для которых $Q(x^0) = 1$ и сумма $Q(x^n a)$ по $a \in A$ равна $Q(x^n)$, или кодовыми условными распределениями вероятностей $\{q(a|x^n),\ a \in A\}$. Для такого кодирования длина получаемого двоичного *кодового слова* меньше $(-\log_2 Q(x^n) + 2)$.

Вероятность появления символа $x_{n+1} = a \in A$ на выходе источника с памятью зависит от *текущего контекста* $s_d = x_n, \ldots, x_{n-d+1} \in A^d$, $d \leq D$, в общем случае переменной длины. Множество всех возможных контекстов можно представить в виде узлов M-ичного дерева глубины D. Контекст (последовательность) u описывает путь из корня дерева в узел, также обозначаемый через u. Любому узлу u соответствует подпоследовательность $x^n(u) \in x^n$ всех символов x_j, для которых индексы j удовлетворяют условию $x_{j-1}, \ldots, x_{j-d} = u,\ d = |u|$.

Обычно истинные условные вероятности неизвестны, и кодовые условные вероятности $q(a|s)$ зависят от характеристик одной или нескольких подпоследовательностей $x^n(u)$. К таким характеристикам относятся: количество $f(a|x^n(u)) = f_n(a|u)$ символов a в $x^n(u)$, алфавит $A(u, x^n) = A_n(u) = \{a: \ f_n(a|u) > 0\}$, его мощность $m_n(u) = |A_n(u)|$ и т.д. Уже при небольших D число встречающихся в сообщении состояний велико, длины подпоследовательностей $x^n(s)$ в среднем малы, а их статистика недостаточна для эффективного сжатия. В частности, затрудняется задача *мультиалфавитного* кодирования, адаптирующегося к неизвестным $A_n(s)$.

Мультиалфавитные свойства алгоритма PPM [1] основаны на (неявном) предположении: чем больше общая (начальная) часть контекстов s, тем больше (в среднем) *близость* их условных распределений вероятностей. Высокая эффективность PPM означает, что это предположение справедливо для большинства кодируемых источников.

Пусть Z_n — множество узлов $s_d,\ d \leq D$, на текущей ветви $x_n, x_{n-1}, \ldots$ контекстного дерева; $s(a)$ — контекст максимальной длины, для которого $f_n(a|s) > 0$; $d_n(a) = |s(a)|$ (если такого узла нет, то $d_n(a) = -1$); $q_n^*(a|s)$ и $q_n^*(esc|s)$ — условные вероятности, используемые для кодирования в узле s символов $a \in A_n(s)$ и символа ухода (escape), описывающего факт появления нового (но неизвестно, какого именно) символа.

Главная особенность всех модификаций алгоритма PPM — представление кодовой условной вероятности любого символа $x_{n+1} \in A$ в виде

$$q(x_{n+1}|x^n) = \left[\prod_{i=d+1}^{D} q_n^*(esc|s_i)\right] q_n^*(x_{n+1}|s_d), \qquad d = |s(x_{n+1})| \,. \tag{1}$$

Произведение условных вероятностей уходов описывает декодеру последовательный спуск из s_D в $s(a)$, где и происходит кодирование (если $d(x_{n+1}) = D$, то это произведение равно 1). Обычно

$$q_n^*(a|s) = \frac{t_n(a|s)}{T_n(s)},\ \forall a \in A_n^*(s); \quad q_n^*(esc|s) = \frac{t_n(esc|s)}{T_n(s)}, \tag{2}$$

где $A_n^*(s) = A_n(s) \setminus A_n(s')$, s' — сын s на текущей ветви Z_n, $t_n(\cdot|s)$ — *обобщенные* частоты символов и символа ухода для $x^n(s)$, $T_n(s)$ — сумма всех обобщенных частот. Как правило, $t(\cdot|s)$ представляются в виде $t(\cdot|s) = f(\cdot|s) + c(\cdot|s)$. Так, PPMC [2] соответствует выбору $c(a|s) = c(esc|s) = 0$; для PPMD [3] $c(a|s) = -1/2,\ c(esc|s) = -m(s)/2$. Наконец, PPMDE [4] отличается от PPMD только выбором $c(esc|s) = -(m(s) + 1/2)/2$.

Во время спуска с помощью символов ухода по текущей контекстной ветви Z_n необходимо исключать из рассмотрения те символы, которые уже проверялись контекстами старших порядков (exclusions). Будем называть их *замаскированными символами*, а контексты, содержащие только такие символы, — *замаскированными контекстами*.

После кодирования очередного символа x_{n+1} необходимо обновить (увеличить на единицу) его частоту в различных узлах $s \in Z_n$. Сначала [1] использовались полные обновления (full updates), при которых частота увеличивается во всех $s \in Z_n$. Позднее [2] были предложены более эффективные частичные обновления (update exclusions), при которых частота обновляется только в узлах s длины $|s| \geq |s(a)|$. В некотором смысле полные и частичные обновления соответствуют двум предельным случаям.

Далее для краткости под словом "контекст s_n" будем понимать не только последовательность символов длины n, но и все собранные об этой цепочке сведения: число и обобщенную частоту различных символов, встречавшихся после этой цепочки, обобщенную частоту символа ухода и т.д. Например, словосочетание "добавление нового символа в контекст s_n" будет означать "добавление информации о новом символе в список всех различных символов, встречавшихся после заданной цепочки s_n". Будем называть контекст s *бинарным*, если $m(s) = 1$ и используются только две условные вероятности — единственного символа и символа ухода, т.е. $q(a|s) \equiv 1 - q(esc|s)$. Под дочерним контекстом s_{d+1} контекста s_d будем понимать любой контекст вида $s_{d+1} = s_d a,\ a \in A$; наоборот, контекст s_d является родительским для контекста s_{d+1}.

§ 3. Выбор обобщенных частот символов

Общее выражение. В настоящей работе предлагается следующее определение обобщенных частот символов:

$$t(a|s_i) = t_0(a|s_i) + \sum \delta_1(a, s_i, s_{i+1}) + \sum \delta_2(a, s_{i-1}, s_i, s_{i+1}), \tag{3}$$

где $s_k = s(a)$, суммирование производится всякий раз, когда контекст s_i принадлежит текущей активной цепочке контекстов ($s_i \in Z_j$). Точность представления частоты равна единице для бинарных контекстов и 1/8 для небинарных. Статистика в небинарном контексте s_i масштабируется (величины всех частот символов и символа ухода делятся на два) всякий раз, когда значение одной из обобщенных частот $t(a|s_i)$ превышает величину 31.

Первое слагаемое вычисляется однократно, после добавления нового символа a в контекст s_i ($t(a|s_i) = 0$). Его значение равно

$$t_0(a|s_i) = \begin{cases} 1 + \dfrac{t(a|s_k) - 1}{T(s_k) - t(a|s_k)}\ , & T(s_i) = 0\ , \\ \\ \dfrac{T(s_i) \cdot t(a|s_k)}{T(s_k) - t(a|s_k) + T(s_i) - m(s_i)}\ , & T(s_i) > 0\ . \end{cases} \tag{4}$$

Слагаемое $\delta_1(a, s_i, s_{i+1})$ вычисляется непосредственно *после* кодирования символа a и равно

$$\delta_1(a, s_i, s_{i+1}) = \begin{cases} 3/4\ , & \text{если } s_i = s(a)\ \cap\ t(a|s_i) < 2\ \cap\ \overline{t(s_i)} < 4\ , \\ 1\ , & \text{если } s_i = s(a)\ \cap\ (t(a|s_i) \geq 2\ \cup\ \overline{t(s_i)} \geq 4)\ , \\ 1/2\ , & \text{если } s_{i+1} = s(a)\ \cap\ t(a|s_{i+1}) < 8\ , \\ 0\ , & \text{в остальных случаях}\ , \end{cases} \tag{5}$$

где $\overline{t(s)} = T(s)/(m(s)+1)$.

Слагаемое $\delta_2(a, s_{i-1}, s_i, s_{i+1})$ вычисляется *перед* кодированием произвольного символа $a = x_{n+1}$ в контексте s_i, если a является наиболее вероятным символом в s_i, и при выполнении условий $(T(s_{i+1}) = 0\ \cap\ T(s_i) \neq 0\ \cap\ T(s_i) < T(s_{i-1})\ \cap\ p(a|s_i) < p(a|s_{i-1}))$. Величина этой поправки выбрана равной

$$\delta_2(a, s_{i-1}, s_i, s_{i+1}) = \frac{T(s_{i-1})}{T(s_i) + T(s_{i-1})} \left(\hat{t}(a|s_{i-1}) - t(a|s_i) \right) \ , \tag{6}$$

где

$$\hat{t}(a|s_{i-1}) = t(a|s_{i-1}) \frac{T(s_i) - t(a|s_i)}{T(s_{i-1}) - t(a|s_{i-1})} \ .$$

Теперь рассмотрим введенные выше правила подробнее.

Механизм наследования информации. В PPM рассматриваются контексты длины $d \le D$. С увеличением D эффективность сжатия начинает уменьшаться. Это объясняется тем, что с ростом порядка модели резко возрастает число контекстов, каждый отдельный контекст встречается слишком редко и избыточность кодирования коротких подпоследовательностей велика. Для исправления этого недостатка PPM можно воспользоваться близостью функций распределения в дочерних и родительских контекстах и задать начальные значения частот символов в дочернем контексте с учетом информации об этом символе, собранной в родительском контексте. Будем называть такой прием *наследованием информации*.

Введем обозначения: s_i — новый контекст ($T(s_i) = 0$) или контекст, к которому добавляется новый символ a ($t(a|s_i) = 0$), $s_k = s(a)$. Будем рассматривать процедуры создания нового контекста и добавления к существующему контексту нового символа раздельно. Локально, в данной точке кодируемого текста, для получения оценки вероятности символа a наиболее оптимальным было бы сразу использовать PPM-модель порядка k, т.е. снизить высоту дерева контекстов до k; тем самым мы исключаем ошибки, связанные с неточностью определения вероятностей ухода. С другой стороны, необходимо учесть, что потомки узла s_k, как правило, имеют более специализированную статистику, и нам нужна статистика, похожая на статистику в создаваемом контексте s_i, поэтому будем производить снижение высоты дерева контекстов только вдоль цепочки контекстов $s_j \ , \quad k < j \le i$. Снижение высоты дерева убирает искажения функции распределения вероятностей символов в узле s_k, внесенные маскировкой символов при частичных обновлениях и повышает точность оценки вероятности символа a в контексте s_k.

Приравняем оценки вероятности символа a в создаваемом контексте и в контексте s_k для дерева с обрезанной контекстной ветвью:

$$\frac{t_0(a|s_i) - 1}{t_0(a|s_i) - 1 + t_0(esc|s_i)} = \frac{t(a|s_k) - 1}{T(s_k) - 1 + T_{i,k}} \ , \tag{7}$$

где

$$T_{i,k} = \sum_{j=k+1}^{i} \left(T(s_j) - t(esc|s_j) - \sum_{n=1}^{m(s_j)} t_0(a_n|s_j) \right) \ ,$$

здесь $T_{i,k}$ — статистика, накопленная в контекстах $s_j, \ k < j \le i$; всюду используются обобщенные частоты символов *до* их изменения после кодирования символа a, а частота символа ухода $t_0(esc|s_i)$ будет определена ниже. Аналогично, при добавлении нового символа в существующий контекст s_i получаем

$$\frac{t_0(a|s_i)}{t_0(a|s_i) + T(s_i)} = \frac{t(a|s_k)}{T(s_k) + T_{i,k}} \ , \tag{8}$$

где используются частоты, взятые уже *после* их обновления символом a.

Формулы (7), (8) можно существенно упростить. Как правило, уход происходит на родительский контекст, т.е. $k = i - 1$; если это не так, то статистика, накопленная в промежуточных контекстах s_j, мала и состоит в основном из унаследованных частот $t_0(\cdot|s_j)$. Поэтому не будем учитывать все промежуточные контексты s_j в сумме $T_{i,k}$. При $T(s_i) = 0$ выберем начальное значение частоты уходов $t_0(esc|s_i)$ по методу PPMC или PPMD (они дают одинаковый результат из-за малой точности описания частот в бинарном контексте). При добавлении символа в контекст заменим $t(esc|s_j)$ и $t_0(a|s_j)$ их оценкой по методу PPMD. Используя эти упрощения в формулах (7), (8) и разрешая их относительно начальных значений обобщенных частот символов $t_0(a|s_i)$, приходим к формуле (4). Полученные формулы (4) достаточно просты и обеспечивают сжатие, в среднем не уступающее сжатию построенному на основе формул (7), (8) без введенных упрощений.

При выводе формулы (4) предполагалось, что $t_0(a|s_i)$ вычисляется сразу после первого появления символа a в контексте s_i. Но при больших D предпочтительнее отложить вычисление до момента, когда s_i встретится в следующий раз; в этом случае распределение символов в новом родительском контексте $s'_k = s'(a)$ обычно ближе к распределению символов в s_i, а вычисленное значение $t_0(a|s_i)$ лучше соответствует неизвестной условной вероятности a в s_i.

Модификация частичных обновлений. В алгоритме PPM, не использующем механизм наследования информации, родительские контексты используются только для кодирования новых символов, не встречавшихся ранее в дочерних контекстах. При введении механизма наследования информации их роль повышается — теперь они служат и для нахождения значений $t_0(a|s_i)$. Поэтому становится более важной скорость накопления собираемой статистики в родительских контекстах. Стандартный механизм частичных обновлений не удовлетворяет этим требованиям. С другой стороны, полные обновления по-прежнему работают неудовлетворительно.

Внесем следующее изменение в обычные частичные обновления: наряду с увеличением на единицу частот $t(a|s_j)$, $j \geq k$, $k = |s(a)|$, будем увеличивать на $1/2$ и частоту в родительском контексте s_{k-1} контекста s_k (строка 3 в формуле (4)). При частичных обновлениях обобщенная частота символа a и его эмпирическая вероятность в родительском контексте пропорциональны числу дочерних контекстов этого контекста, в которых встречался символ a. Чтобы не потерять полностью это свойство в предложенной модификации, будем прекращать обновление контекста $t(a|s_{k-1})$ после достижения частоты символа в контексте $s(a)$ некоторого порогового значения. Экспериментально оно выбрано равным 8.

Улучшение оценок вероятностей наиболее вероятного символа и наименее вероятных символов. Поскольку статистика в родительских контекстах накапливается быстрее, чем в "молодых" контекстах-потомках (с малыми $T(s)$), иногда целесообразно неоднократное обращение к статистике родительского контекста.

В PPMII обобщенная частота наиболее вероятного символа a_1 корректируется с помощью информации о символе из родительского контекста, если дочерний контекст не набрал еще достаточной статистики ($T(s_i) < T(s_{i-1})$) и выдает заниженную оценку вероятности наиболее вероятного символа по сравнению с родительским контекстом ($q(a_1|s_i) < q(a_1|s_{i-1})$). Новое значение $t_1(a_1|s_i)$ обобщенной частоты находится с помощью взвешивания значения частоты символа в текущем контексте и приведенного значения $\hat{t}(a_1|s_{i-1})$ частоты в родительском контексте (см. (6)). Если родительский контекст является бинарным, то используется бинарный родительский контекст в котором обобщенная частота символа a_1 максимальна.

Неточная оценка частоты маловероятного символа a_n (для оценки маловероятности символа используется условие ($t(a_n|s_i) < 2$)) может оказать большое влияние на оценку вероятности высоковероятных символов в молодых контекстах (условие для молодого контекста: ($\overline{t(s_i)} < 4$), где $\overline{t(s)} = T(s)/(m(s)+1)$ — среднее значение частоты символа в контексте, включая символ ухода). Поэтому частоты таких символов просто увеличиваются на $3/4$, а не на единицу (строка 1 в формуле (5)).

§ 4. Адаптивная оценка вероятности и обобщенных частот ухода

Для определения вероятности ухода разобьем все контексты на три типа: бинарные контексты, небинарные контексты без замаскированных символов (см. § 2), небинарные контексты с замаскированными символами. Рассмотрим каждый тип контекстов отдельно.

Оценка вероятности ухода с бинарного контекста. Если D достаточно велико, то для текстовых файлов в 70–80% случаев кодирование символа начинается с бинарного контекста. Для таких контекстов кодовая вероятность единственного символа a однозначно связана с вероятностью ухода: $q(a|s) = 1 - q(esc|s)$. Поэтому уточнение вероятности ухода для этого случая дает заметный выигрыш в эффективности сжатия. Для получения ее оценки построим дополнительную модель, оценивающую вероятность ухода с контекста s в зависимости от некоторого набора параметров $w(s) = (w_1, \ldots, w_h)$, связанных с текущим контекстом. Для того чтобы не путать эту модель с основной моделью, будем называть ее *SEE (secondary escape estimation) моделью* [5], а параметры этой модели — *SEE-контекстами.* Оценка вероятности ухода в каждом SEE-контексте вычисляется по обычной формуле среднего значения

$$q(esc|w) = \frac{1}{N(w)} \sum_{j=1}^{N(w)} \delta_j(w) \ , \tag{9}$$

где $\delta_j = 1$, если на j-м шаге появился новый символ (произошел уход на родительский контекст), и $\delta_j = 0$ в противном случае, а $N(w)$ — число испытаний.

Для повышения адаптивности оценки среднего значения в алгоритме PPMII применяется схема с непрерывным масштабированием статистики. Общее число испытаний N формально фиксируется: $N = N_0$, $(1 - 1/N_0)$ является масштабирующим множителем. На каждом шаге к сумме $S = \sum \delta_j$ добавляется величина δ_j и вычитается среднее значение $\langle \delta_j \rangle$ ($\langle \delta_j \rangle = S/N_0$), т.е. изменение суммы S на каждом шаге составляет $\Delta S = \delta_j - S/N_0$. Для замены деления двоичными сдвигами величина N_0 выбирается равной степени двойки.

Число компонент h вектора $w(s)$ и их упорядочение по степени влияния на величину $q(esc|w)$ может быть определено только экспериментально. Ниже перечисляются компоненты, использованные в PPMII (в порядке убывания степени влияния).

1) В наибольшей степени вероятность ухода коррелирует с обобщенной частотой единственного символа в контексте s_i, поэтому эта величина выбрана в качестве элемента SEE-контекста; квантизуем ее до 128 значений.

2) Алгоритм PPM использует сходство родительских и дочерних контекстов, поэтому число $m(s_{i-1})$ различных символов, встречавшихся в родительском контексте, сильно влияет на вероятность ухода с дочернего контекста s_i. Добавим число символов в родительском контексте, квантизованное до восьми значений, к SEE-контексту.

3) Эксперименты показывают, что на реальных данных блоки хорошо предсказуемых данных чередуются с блоками плохо предсказуемых данных. Величина таких блоков невелика — порядка 3–5 символов, что соответствует разделению на слова и части слов в текстах на естественных языках. Для того чтобы отслеживать переключение между такими блоками, вставим в SEE-контекст вероятность появления предыдущего символа в предыдущем контексте, квантизованную до двух значений.

4) *Длинным блоком символов длины* L назовем последовательность символов входного текста, для которой при кодировании ни разу не происходил уход на родительские контексты, и L символам из нее были присвоены кодовые вероятности больше, чем $1/2$. Вполне вероятно, что порядок PPM-модели $D < L$ будет недостаточным для таких блоков. Чтобы отслеживать подобную ситуацию, вставим в SEE-контекст флажок (два значения), сигнализирующий о том, находится ли кодер в длинном блоке или нет.

5) Естественно, что текущий кодируемый символ в наибольшей степени коррелирует с предыдущим символом. В силу этого желательно было бы ввести в SEE-контекст зависимость от последнего закодированного символа. Однако вставка всего символа целиком в SEE-контекст приведет к большим расходам памяти и малой частоте появления каждого отдельного SEE-контекста. Поэтому добавим в SEE-контекст флажок, принимающий значение 0, если старшие два бита двоичного представления предыдущего символа равны нулю, и 1 — в противоположном случае. Аналогичный флажок добавим также и для двух старших бит предсказываемого символа. Такой прием вводит зависимость SEE-контекста от порядка символов во входном алфавите; с другой стороны, он может рассматриваться как некоторая странная форма хеширования SEE-контекстов.

Итого, SEE-модель для бинарных контекстов состоит из $128 \times 8 \times 2 \times 2 \times 4 = 16384$ SEE-контекстов. Дальнейшее увеличение числа SEE-контекстов не приведет к увеличению степени сжатия, так как большинство SEE-контекстов будет появляться слишком редко, и статистика в них будет недостаточной. Детали квантизации различных элементов SEE-контекста можно найти в исходных текстах программы [6].

Оценка обобщенной частоты ухода для небинарного контекста (без замаскированных символов). Этот тип контекстов представляет собой более сложный случай — с одной стороны, такие контексты встречаются относительно редко в моделях высоких порядков, и для них трудно набрать достаточную статистику, с другой стороны, простые адаптивные методы оценки вероятности ухода, подобные тому, что описан в предыдущем пункте, не дают заметного выигрыша. По этой причине будем использовать полуадаптивный метод, выбрав в качестве начального приближения вероятность ухода по методу PPMD.

Будем считать, что вероятности символов в контексте распределены по экспоненциальному закону, т.е.

$$q(a_n|s_i) = \rho^n(1-\rho) \ , \quad n = 0, 1, 2, \dots \tag{10}$$

Тогда при превращении бинарного контекста в небинарный, на основе результатов предыдущего пункта можно найти основание степени ρ, и из него найти значение частоты уходов $t_0(esc|s_i)$ для контекста, содержащего два символа. Эти вычисления можно произвести только численно, так как символы необязательно появляются в контексте в порядке убывания вероятности.

Далее значение $t(esc|s_i)$ будем изменять только при добавлении новых символов в контекст, аналогично тому, как это делается в методе PPMD, но не для каждого добавляемого символа. Будем добавлять к частоте $t(esc|s_i)$ величину

$$\delta_1 = 1/2 \qquad \text{при} \quad 2m(s_i) < m(s(a)), \tag{11}$$

где $m(s)$ — число символов в контексте s. Дополнительная корректировка требуется, когда новый добавленный символ a имеет малую вероятность в текущем контексте; добавим в этом случае величину

$$\delta_2 = 1 - t_0(a|s_i) \quad \text{при} \quad t_0(a|s_i) < 1\ . \tag{12}$$

Итоговая формула такова:

$$t(esc|s_i) = t_0(esc|s_i) + \sum \left(\delta_1(s_i, s(a)) + \delta_2(a, s_i)\right)\ , \tag{13}$$

где $t_0(esc|s_i)$ вычисляется при превращении бинарного контекста в небинарный, суммирование происходит всякий раз, когда в контексте s_i встретится новый символ, а δ_1 и δ_2 даются формулами (11) и (12) соответственно.

Оценка обобщенной частоты ухода для контекста с замаскированными символами. Для этого типа контекстов вероятность ухода в наибольшей степени зависит от суммарной частоты символов в контексте. Эта величина должна быть представлена с весьма высокой точностью, что приведет к редкому появлению отдельных SEE-контекстов и, соответственно, к низкой точности оценки вероятности ухода. Поэтому будем моделировать поведение не вероятности ухода, а частоты уходов, которые слабо зависят от суммарной частоты символов в контексте. Оценка среднего значения частоты уходов строится аналогично оценке среднего значения вероятности ухода для бинарных контекстов.

Частота уходов сильнее всего зависит от числа незамаскированных символов (квантизуем эту величину до 42 значений). Далее опять воспользуемся сходством поведения родительских и дочерних контекстов, и будем сравнивать число незамаскированных символов в текущем контексте с числом символов в дочернем контексте и с числом символов, которые будут незамаскированы в родительском контексте в случае, если текущий контекст не сможет обработать сжимаемый символ. Это добавляет еще четыре значения в SEE-контекст. По аналогии с бинарными SEE-контекстами введем зависимость от старших двух бит предыдущего символа (два значения). Наблюдается также корреляция частоты ухода с отношением $\overline{t(s)} = T(s)/(m(s)+1)$, т.е. со средним значением частот символов в контексте. Эта величина квантизуется до двух значений.

Итого общее число SEE-контекстов: $42 \times 4 \times 2 \times 2 = 672$.

§ 5. Упрощенный PPMII

Описанный в §§ 3, 4 алгоритм обеспечивает высокую эффективность сжатия, но сложен в реализации. Поэтому ниже рассматриваются его упрощения, сохраняющие достаточно высокую степень сжатия; одновременно обеспечивается сложность реализации, сравнимая с наиболее распространенными практическими программами, основанными на алгоритмах LZ77 и BWT .

Для механизма наследования информации, введенного в § 3, будем вычислять обобщенные частоты сразу после добавления нового символа в контекст. Тем самым исключается один дополнительный поиск и просмотр родительского контекста. Модификация частичных обновлений, описанная в том-же параграфе, требует одного дополнительного сканирования родительского контекста на каждый шаг алгоритма и может увеличивать время выполнения почти вдвое, что неприемлемо. Можно предположить, что если символ был обработан контекстом наибольшего порядка ($|s(a)| = D$), то статистика в данной цепочке контекстов уже устоялась, вероятность использования родительского контекста мала, и нет необходимости обновлять его статистику. При использовании этого предположения время выполнения увеличивается всего лишь на 2–10% в зависимости от значения D. Улучшение оценки вероятности наиболее вероятного и наименее вероятных символов (§ 3) требует громоздких вычислений и дает слишком малый выигрыш по эффективности сжатия, поэтому в упрощенной модели оно не производится.

Для адаптивной оценки вероятности ухода с бинарного контекста используется более простой и менее адаптивный способ подсчета среднего значения величин (9). Из-за малой скорости адаптации статистика в каждом отдельном SEE-контексте будет недостаточной для коротких файлов, поэтому необходимо уменьшить число SEE-контекстов. Для бинарных контекстов отключим зависимость SEE-контекста от следующих компонент: флажок, показывающий, находится ли кодер в длинном блоке символов или нет, и флажки, составленные из старших бит предыдущего и предсказываемого символа. Для контекста с замаскированными символами отключим флажок, составленный из старших бит предыдущего символа. Оценка обобщенной частоты ухода с небинарного контекста остается неизмененной.

Точность представления частот символов для упрощенной модели понижена с 1/8 до 1/4.

§ 6. Результаты экспериментов

Описанный в данной статье алгоритм PPMII и его упрощенная модификация были реализованы на языке программирования C++ и публично доступны по адресу [6]. Основному алгоритму PPMII соответствует исполнимый файл `PPMonstr.exe`, а упрощенному алгоритму — `PPMd.exe`. Все эксперименты проводились на стандартном наборе файлов Calgary Corpus [7].

Сравнение эффективности сжатия. В первом эксперименте сравнивается эффективность PPMII с наилучшими из описанных в литературе схем сжатия. В табл. 1 представлены результаты (в битах на байт) работы следующих схем сжатия: реализации [8] метода CTW [9] с двоичной декомпозицией символов (результаты взяты из работы [10]), ассоциативного кодера Г. Буяновского (ACB) [11], лучшего из описанных С. Бантон FSMX-кодеров [12], PPMZ2 Ч. Блума [13]. В трех последних колонках даны результаты PPMII для D = 5, 8 и 16. Необходимо отметить, что реализация [8] метода CTW использует разбиение символов на биты, специально оптимизированное для английских текстов.

Алгоритм PPMII дает лучшие результаты на всех файлах, кроме являются файлы `book1`, `obj1` и `pic`. На файле `book1` реализация CTW [8] дает лучшие результаты, так как она специально оптимизирована под подобный тип данных. Программа ACB использует бит-ориентированный метод сжатия, поэтому она набирает статистику примерно в восемь раз быстрее, чем байт-ориентированные программы. Это дает ей преимущество на маленьких файлах, что видно на примере файла `obj1`, самого короткого в Calgary Corpus. В файле `pic` записано изображение с глубиной цвета 1 бит, и естественно, что такие данные лучше всего моделируются бит-ориентированными методами, например, ACB.

Таблица 1

Эффективность различных схем сжатия

Файл	CTW	ACB	FSMX	PPMZ2	PPMII - 5	PPMII - 8	PPMII - 16
`Bib`	1,782	1,935	1,786	1,717	1,760	1,717	**1,712**
`Book1`	**2,158**	2,317	2,184	2,195	2,181	2,169	2,174
`Book2`	1,869	1,936	1,862	1,846	1,874	1,821	**1,816**
`Geo`	4,608	4,555	4,458	4,576	4,332	4,326	**4,322**
`News`	2,322	2,317	2,285	2,205	2,240	2,188	**2,175**
`Obj1`	3,814	**3,498**	3,678	3,661	3,545	3,542	3,540
`Obj2`	2,473	2,201	2,283	2,241	2,286	2,198	**2,160**
`Paper1`	2,247	2,343	2,250	2,214	2,211	2,179	**2,173**
`Paper2`	2,190	2,337	2,213	2,185	2,179	2,161	**2,158**
`Pic`	0,800	**0,745**	0,781	0,751	0,765	0,751	0,749
`Progc`	2,330	2,332	2,291	2,258	2,249	2,203	**2,189**
`Progl`	1,595	1,505	1,545	1,445	1,585	1,461	**1,423**
`Progp`	1,636	1,502	1,531	1,450	1,608	1,515	**1,439**
`Trans`	1,394	1,293	1,325	1,214	1,377	1,252	**1,210**
Средний bpb	2,230	2,201	2,177	2,140	2,157	2,106	**2,089**

Сравнение затрат памяти и процессорного времени. Во втором эксперименте сравнивались интегральные характеристики программной реализации PPMII и упрощенного PPMII с наиболее распространенными практическими реализациями алгоритмов LZ77 (ZIP [14]), BWT (BZIP2 [15]), а также с наиболее мощной реализацией алгоритма PPM* (PPMZ2 [13]).

Таблица 2

Интегральные характеристики различных программ сжатия

Порядок модели D	Упрощенный PPMII			PPMII		
	Средний bpb	Время,сек.	Память	Средний bpb	Время,сек.	Память
2	2,758	3,24	0,5	2,763	4,94	0,6
3	2,369	3,84	1,2	2,376	6,04	1,3
4	2,219	4,56	2,2	2,221	7,14	2,3
5	2,168	5,38	4,1	2,157	8,35	4,2
6	2,150	6,26	7,8	2,129	9,50	7,9
8	2,134	7,74	18,7	2,106	11,42	18,8
10	2,129	9,00	33,1	2,097	12,02	33,2
16	2,123	*	83,7	2,089	*	83,8
Для сравнения						
ZIP -9	2,764	5,93	0,5			
BZIP2 -8	2,368	5,87	6,8			
PPMZ2	2,140	*	> 100			

* Измерение времени выполнения не производилось из-за недостатка оперативной памяти.

Все программы за исключением PPMZ2 были оттранслированы одним и тем же компилятором (Intel C v.4.0) с одинаковыми настройками компиляции. Эксперименты проводились на компьютере PII–233 (292) МГц с оперативной памятью 64 МБ под ОС Windows98 и файловой системой FAT32. Измерение времени производилось системным таймером, каждая программа запускалась пять раз, и выбиралось наименьшее время выполнения. Результаты измерения средней невзвешенной скорости сжатия (в битах на байт), времени сжатия всего набора файлов (в секундах) и пиковой потребности в памяти (в мегабайтах) представлены в табл. 2.

Из этой таблицы видно, что упрощенный вариант PPMII обеспечивает широкий круг возможностей. При малых D (2–3) он дает степень сжатия, сравнимую с ZIP и BZIP2, при большем быстродействии и меньших затратах памяти, чем у BZIP2. При средних значениях D (4–6) он дает заметно лучшую степень сжатия; затраты времени и памяти, при этом остаются сравнимыми с BZIP2. И наконец, при D=8–16 он превосходит лучшую из описанных до сих пор программу PPMZ2 по степени сжатия, быстродействию и требованиям к памяти.

Основная модификация алгоритма PPMII, как правило, дает несколько лучший результат, чем упрощенный вариант. Исключением являются только малые D (2–4). Это объясняется тем, что при малых D число бинарных контекстов невелико, и статистика в них не успевает устояться; так как основной вариант использует более широкий SEE-контекст, то на нем это сказывается сильнее.

Автор благодарит Ю.М. Штарькова за помощь в написании этой статьи.

Список литературы

[1] *Cleary J.G. and Witten I.H.* Data Compression Using Adaptive Coding and Partial String Matching, // IEEE Trans. Commun. 1984. V. 32. № 4. P. 396–402.

[2] *Moffat A.* Implementing the PPM Data Compression Scheme, // IEEE Trans. Commun. 1990. V. 38. № 11. P. 1917–1921.

[3] *Howard P. G.* The design and analysis of efficient lossless data compression systems, PhD thesis. Brown University, 1993.

[4] *Aberg J., Shtarkov Yu.M. and Smeets B.J.M.* Multialphabet coding with separate alphabet description, // Proc. of Compression and Complexity of Sequences 97, Positano, Salerno, Italy, IEEE Comp. Soc. Press, 1998. P. 56–65.

[5] *Bloom C.* Solving the Problems of Context Modeling, 1996. `www.cbloom.com/papers/`.

[6] *Shkarin D.* PPMd — fast PPM compressor for textual data, 2000. `ftp.elf.stuba.sk/pub/pc/pack/ppmdf.rar`. E-mail: `shkarin@arstel.ru`

[7] The Calgary Compression Corpus. `ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus/`

[8] *Volf P.A.J.* Text compression methods based on context weighting, // Technical report. Stan Ackermans Institute, Eindhoven University of Technology, 1996.

[9] *Willems F., Shtarkov Y. and Tjalkens T.* The Context-Tree Weighting Method: Basic Properties, // IEEE Trans. Inform. Theory, 1995. V. 41 № 3. P. 653–664.

[10] *Volf P.A.J and Willems F.M.J.* Switching between two universal source coding algorithms, // Proc. IEEE Data Compression Conf. 1998. P. 491–500.

[11] *Буяновский Г.* Ассоциативное кодирование, // Монитор, 1994. № 8. с.10–22.

[12] *Bunton S.* On-line stochastic processes in data compression, PhD thesis, Washington, 1996.

[13] *Bloom C.* PPMZ2 — High Compression Markov Predictive Coder, 1999. `www.cbloom.com/src/`.

[14] *Info-ZIP Group.* Zip v.2.3 — compression and file packaging utility, 1999. `www.cdrom.com/pub/infozip/`.

[15] *Seward J.* BZip2 v.1.0 — block-sorting file compressor, 2000. `www.muraroa.demon.co.uk/`.